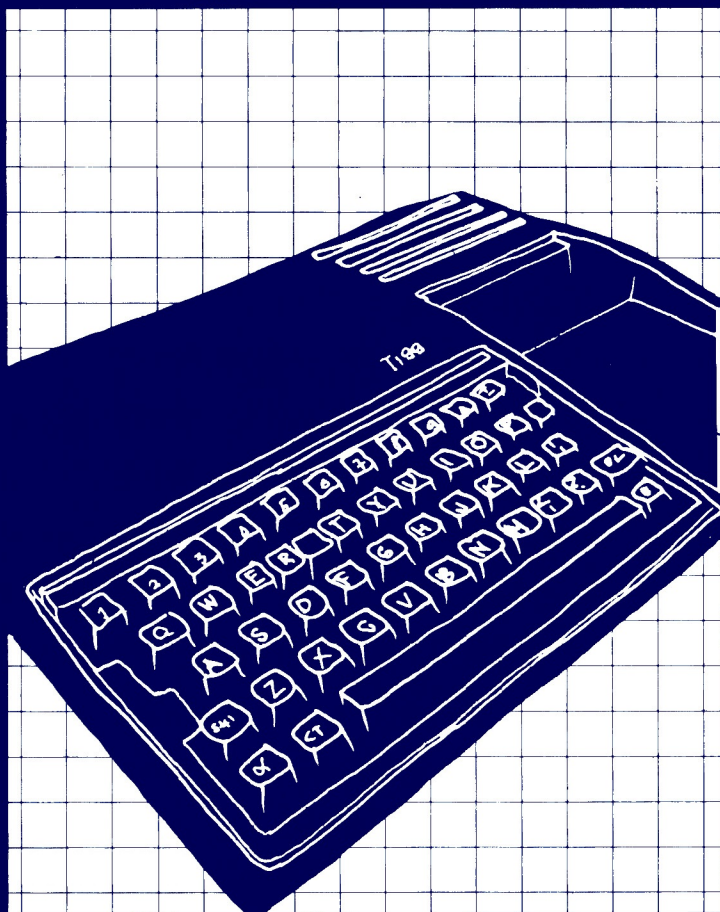
seconda edizione

Herbert D. Peckham

Imparate il Basic con il Texas TI 99/4A

Il piacere del computer

Il piacere del computer serie diretta da Mauro Boscarol

1 *Tom Rugg e Phil Feldman* 32 programmi con il PET
2 *Rich Didday* Intervista sul personal computer, hardware
3 *Tom Rugg e Phil Feldman* 32 programmi con l'Apple
4 *Ken Knecht* Microsoft Basic
5 *Paul M. Chirlian* Pascal
6 *Tom Rugg e Phil Feldman* 32 programmi con il TRS-80
7 *Rich Didday* Intervista sul personal computer, software
8 *Herbert D. Peckham* Imparate il Basic con il PET/CBM
9 *Karl Townsend e Merl Miller* Il personal computer come professione
10 *Karen Billings e David Moursund* Te ne intendi di computer?
11 *Thomas Dwyer e Margot Critchfield* Il Basic e il personal computer, uno: introduzione
12 *Don Inman e Kurt Inman* Imparate il linguaggio dell'Apple
13 *Thomas Dwyer e Margot Critchfield* Il Basic e il personal computer, due: applicazioni
14 *Luigi Pierro* Il manuale del CP/M
15 *Carlo Sintini* A scuola con il PET/CBM
16 *David-Johnson-Davies* Il manuale dell'Atom
17 *David E. Schultz* Il libro del Commodore VIC 20
18 *Him Huffman e Robert Bruce* Il "debug" nei personal computer
19 *John M. Nevison* Programmazione in Basic per l'uomo d'affari
20 *Mark Harrison* Imparate il Basic con lo ZX81
21 *Ronald W. Andersen* Dal Basic al Pascal
22 *Herbert D. Peckham* Imparate il Basic con il Texas TI 99/4A

Herbert D. Peckham

Imparate il Basic con il Texas TI 99/4A

franco muzzio & c. editore

Indice generale

Prefazione

Questo libro è una rielaborazione di un precedente lavoro dell'autore già pubblicato dalla McGraw-Hill Book Company. Il libro, intitolato *Basic: A Hands-On Method*, introduce gli studenti al Basic con riferimenti a diversi computer operanti in *time sharing*. Il contenuto di questo libro è stato poi modificato e adattato in modo specifico all'home computer della Texas Instruments Incorporated. Poiché le motivazioni e le idee che sono state alla base della opera originale sono ugualmente valide rispetto il TI home computer, esse sono state riprese in questa sede.
Due caratteristiche della maggior parte dei testi sulla programmazione in Basic in commercio sono molto criticabili. Primo, quasi tutti iniziano subito ad usare una matematica ad un livello che esclude gran parte delle persone a cui noi siamo estremamente interessati, molte delle quali conoscono solo i principi dell'algebra (che magari non ricordano chiaramente) ma che, per vari motivi, intendono imparare a programmare in Basic. La seconda obiezione è che generalmente questi testi non richiedono al principiante di trascorrere molto tempo sul calcolatore (a volte non ne richiedono per niente). Solitamente gli studenti tentano di imparare la programmazione come una qualsiasi altra materia e non sentono il bisogno di sperimentare ed eseguire programmi sul computer. Sembra assiomatico che l'apprendimento della programmazione è molto più efficace se buona parte dello studio del Basic si fa utilizzando il calcolatore.
La tesi principale di questo testo è che il materiale tradizionale deve essere preceduto da una buona quantità di tempo trascorsa usando il lin-

guaggio del computer. L'esperienza acquisita fino ad ora conferma l'ipotesi che gli studenti riescono ad apprendere l'argomento più rapidamente e con maggiore efficacia con questa iniziale esposizione del Basic sul computer.
Molti libri di testo sono usati nel contesto scolastico come parte del sistema educativo formale. Certamente molti studenti impareranno la programmazione in questo ambiente tradizionale. Tuttavia, la vendita di computer per uso familiare tocca tutti i settori della nostra società. Questo significa che l'usuale concetto di "studente" deve essere cambiato radicalmente. Questo testo è stato concepito per essere utile a qualsiasi persona (che faccia o no parte di un sistema educativo) che voglia imparare come programmare il TI home computer.
Il lettore noterà subito che il libro è strutturato piuttosto diversamente da molti testi di programmazione. Ogni capitolo inizia con la spiegazione degli argomenti del capitolo stesso. Poi lo studente è guidato nello svolgimento di vari esercizi al calcolatore, che illustrano come agisce il Basic e permettono di familiarizzare con le sue caratteristiche. Quando si è acquisita un'"idea" del Basic, si può procedere con più profitto sul testo tradizionale. Intenzionalmente la matematica è stata utilizzata ad un livello piuttosto basso. Lo studente che ha una conoscenza più approfondita della matematica, non avrà molta difficoltà nel capire come utilizzarla sul computer. Tuttavia, se il livello di matematica nel testo fosse stato troppo alto, la maggior parte dei principianti si sarebbe scoraggiata già ai primi capitoli. Con questo libro quasi tutti dovrebbero essere in grado di imparare senza sentirsi bloccati dalla matematica. Per usare questo testo è necessario che lo studente possa accedere ad un home computer della Texas Instruments.
Il testo è organizzato in undici capitoli. Se utilizzato per l'insegnamento in una classe, ogni capitolo dovrebbe richiedere circa due ore di tempo e possibilmente altre tre o quattro ore nell'orario extrascolastico. Alla fine di ogni capitolo si trovano degli esercizi che permettono allo studente di verificare il suo grado di preparazione. Sono stati introdotti problemi per far acquisire pratica nella programmazione. La soluzione dei problemi con numero dispari viene data alla fine del libro.
Il testo può essere usato in parecchi modi differenti. Il primo, e probabilmente il più importante, è quello di utilizzarlo senza insegnante, come testo di autoapprendimento. È anche stato usato in un accostamento e controllo finale liberi, e corso autogestito. Se lo si desidera, si può presentare il materiale nella forma tradizionale.
Studenti di ogni livello, dalle scuole medie all'università, dalla casalinga alla persona più altolocata, dall'operaio al professionista, saranno in grado di utilizzare il libro senza difficoltà. Lo scopo è di insegnare a programmare in Basic il più rapidamente ed efficacemente possibile. Alcune possibilità del TI home computer non sono state prese in conside-

razione in questo libro. Alcune di queste implicano l'uso della matematica ad un livello superiore a quello assunto nella presentazione. Come indicato sopra, la matematica richiesta non va oltre le prime nozioni di algebra, e l'algebra è usata principalmente per la valutazione di formule. Una migliore conoscenza della matematica può essere utile, ma non necessaria.

Due testi forniti con il TI home computer sono consoni al contenuto e allo stile di questo libro. Per prima cosa c'è il manuale che contiene tutte le funzioni e le possibilità del Basic implementato nel TI home computer. Poche persone, però, saranno in grado di utilizzare queste informazioni inizialmente. In secondo luogo c'è un testo elementare (*Beginners Basic*) che fa conoscere in breve tempo le operazioni del computer e gli elementi della programmazione in Basic.

Dopo aver preso familiarità con il contenuto di questo opuscolo, molti troveranno piacevole scrivere programmi e useranno il manuale per ottenere una risposta ai problemi che emergono di volta in volta. Tuttavia si ritiene che la maggior parte dei principianti avverte uno stacco tra il testo elementare e il manuale. Lo scopo di questo libro è quello di saldare questa frattura. Di conseguenza, gli argomenti sono sviluppati molto gradualmente; se sei una persona molto portata per il computer, troverai la presentazione un po' lenta. D'altro canto, se sei un po' preoccupato all'idea di dover imparare a programmare un computer, a maggior ragione apprezzerai la facile andatura e sarai in grado di apprendere i contenuti senza difficoltà.

Ringraziamenti

La Texas Instruments Inc. ha offerto una generosa assistenza nella stesura di questo libro. Un particolare ringraziamento va a Mr. Alfred Riccomi, Mr. Charles Watkins e Ms. Susan Naff, dipendenti della Texas Instruments, e a Mr. Robert O'Dell che ha letto l'intero manoscritto apportandogli utili revisioni.

Gli errori che rimangono sono naturalmente a me dovuti. Saranno apprezzati commenti e suggerimenti per il miglioramento di questo libro.

HERBERT D. PECKHAM

CAPITOLO UNO

L'home computer Texas Instruments e il Basic

I calcolatori sono ora una parte comune della nostra vita. Anche se non li vediamo, ci sono, ed intervengono in qualche modo in buona parte delle nostre attività quotidiane. Affari di qualunque dimensione, istituti educativi, varie attività del governo: nessuno sarebbe in grado di maneggiare la sbalorditiva quantità di informazioni che sembra caratterizzare la nostra società senza utilizzare dei computer. Solo recentemente, tuttavia, è stato possibile portare dei piccoli calcolatori, poco costosi, nelle case e nelle scuole. Per la prima volta, persone di ogni tipo, da studenti a persone anziane, stanno cominciando ad entrare nel mondo dei computer. Dato che i prezzi dei calcolatori continuano a scendere, questo fatto è destinato ad evidenziarsi. Sempre più persone vorranno sapere come utilizzare i calcolatori, per essere in grado di partecipare pienamente al progredire della nostra società.

CHE COS'È IL BASIC

State per intraprendere lo studio di un linguaggio per calcolatori chiamato Basic, usando un personal computer molto potente che si chiama Texas. Il Basic è un linguaggio molto specializzato, designato a permettere che voi ed il computer vi capiate e possiate comunicare l'uno con l'altro. Questo linguaggio è sicuramente più facile da usare di un linguaggio

parlato, come lo spagnolo o il francese. Il Basic possiede un semplice vocabolario consistente in poche parole, una struttura grammaticale e delle regole di utilizzazione proprio come un qualsiasi altro linguaggio. Il primo compito sarà quello di imparare il vocabolario del Basic e di familiarizzare con le sue regole grammaticali. Vedremo poi in che modo questo linguaggio permette di usare il computer in una grande varietà di applicazioni. Il livello della matematica utilizzata è stato intenzionalmente tenuto molto basso. Perciò se le vostre conoscenze matematiche non sono molte buone, non preoccupatevi. Man mano che procederemo col Basic, avrete modo di rivedere un po' di matematica elementare. Un modo molto efficace di apprendere è quello di osservare i dettagli e le caratteristiche del lavoro che si sta eseguendo: è il metodo di "apprendimento col calcolatore". Questa è la soluzione adottata nel libro. Vi sarà chiesto di iniziare ogni capitolo con degli esercizi pratici sul computer. Dopo aver eseguito le indicazioni ed aver osservato che cosa fa il calcolatore in risposta alle vostre istruzioni, comincerete ad acquisire un'"idea" del Basic e di come opera il computer. Capito questo, potrete procedere con più profitto nello studio del materiale scritto, che riassume quello che avrete imparato.
Quindi, per come viene presentato il Basic in questo libro, gli esercizi guidati sul computer sono una parte chiave dell'apprendimento.

DOV'È NATO IL BASIC?

La versione originale del Basic fu ideata e scritta al Dartmouth College sotto la direzione dei professori John G. Kemeny e Thomas E. Kurtz. Nel settembre del 1963, iniziarono i lavori sul concetto di computer operante in *time sharing* e per la creazione di un linguaggio di programmazione rivolto all'utente. Un'informazione molto interessante è che buona parte dell'effettiva programmazione relativa al progetto fu fatta da studenti di Dartmouth non laureati. La data di nascita del Basic è l'1 maggio 1964, cosicché il linguaggio non è ancora maggiorenne.
Il successo di questo sforzo pionieristico a Dartmouth attirò subito l'attenzione nazionale, e molto presto se ne interessarono altre istituzioni. Il resto è storia. Oggi quasi ogni computer che lavora in *time sharing* è fornito di Basic. I più recenti sviluppi sono dati dall'implementazione del Basic su piccoli home computer. Ogni anno, la percentuale di tutte le attività, svolte con i calcolatori in Basic, aumenta. Quello che iniziò come progetto in un singolo *college*, è ora parte affermata dell'industria dei computer in tutto il mondo.

CHE COS'È L'HOME COMPUTER TEXAS INSTRUMENTS?

Il concetto di computer potente, comparabile per dimensioni e prezzo con un televisore a colori, capace di eseguire quasi tutti i compiti che prima richiedevano grossi calcolatori in ambienti con aria condizionata, è un'idea nuova e piuttosto sconvolgente. Tuttavia, per quanto sconcertante possa essere, questo è precisamente quello che è sucesso. L'home computer prodotto dalla Texas Instruments sembra rappresentare la maggior forza nel cambiare il tradizionale modo di pensare sui computer e sul modo di usarli. Prima di iniziare lo studio della programmazione in Basic sul vostro TI home computer ci fermeremo un attimo ad esaminare le sue origini ed indicheremo alcune delle sue notevoli caratteristiche.

Due cose sono particolarmente importanti riguardo al TI home computer. In primo luogo il prezzo è alla portata di un gran numero di persone, che potranno acquistare o accedere a uno di essi. Questo implica il secondo punto che va messo in rilievo. La questione dell'accessibilità alla potenza di un computer è stata sempre difficile da trattare. Spesso sembrava che delle barriere, alcune reali ed altre immaginarie, fossero poste sulla strada di chi desiderava utilizzare i calcolatori. Con il TI home computer queste barriere non esistono più. Quindi questi nuovi personal computer si possono trovare nelle case, negli uffici e nelle scuole di tutto il paese. Per definizione "elaborazione personale" deve significare "elaborazione accessibile", e la caratteristica del TI è di fornire questa possibilità.

Il cuore dell'home computer è un microcomputer su un *chip* (scaglia). I primi microcomputer di questo tipo furono costruiti nel 1973; la tecnologia utilizzata è quindi molto recente.

Nel TI sono state introdotte alcune caratteristiche estremamente importanti. Il computer si collega tramite un TV color, il che significa che può essere utilizzato il colore e il suono. Sotto il controllo di un programma Basic si possono generare disegni e grafici (in sedici colori) sul monitor TV accompagnati da musiche e rumori. Il vecchio proverbio cinese che "un disegno è meglio di mille parole" è certamente applicabile qui e tocca una delle più potenti caratteritiche del TI home computer. Possiamo aggiungere che "un disegno accompagnato dalla musica è meglio di un milione di parole".

Era chiaro inoltre che il Basic sarebbe stato il linguaggio dell'home computer della Texas Instruments e che avrebbe dovuto essere abbastanza potente da permettere una gran varietà di impieghi, dal principiante che si diverte con dei giochi al programmatore professionista. Il Basic fu scelto per via di precedenti buone esperienze con calcolatori che lavorano in *time sharing*. È un linguaggio "amichevole", che non spaventa,

facile da apprendere ma con ampie possibilità. In breve, è l'ideale per l'elaborazione personale, ed è stato quasi universalmente adottato per i personal computer.
Per molti anni la Texas Instruments è stata all'avanguardia nella tecnologia elettronica. L'apparire del TI home computer è la logica conseguenza di questa leadership. Sostenuta da anni di esperienza la Texas garantisce una assistenza continua ed efficace per il vostro home computer.

COME COMINCIARE

Dovrete affrontare ogni capitolo del libro allo stesso modo. Il materiale è stato organizzato pensando a degli speciali schemi di apprendimento, e un qualsiasi cambiamento sarà meno efficace e vi richiederà più tempo. Ogni capitolo inizia con una breve spiegazione degli argomenti che contiene. Questa parte deve essere studiata attentamente per potere avere un quadro chiaro di quello che sarà fatto. (È bello sapere dove si sta andando!) Quando vi sarà richiesto dovrete segnare le risposte del computer nello spazio apposito. Occasionalmente dovrete rispondere a delle domande. Lo scopo di questi esercizi pratici è di permettervi di farvi un'idea dei concetti usati e di vedere il Basic al lavoro. È importante che voi pensiate a quello che succede nelle situazioni che verranno proposte. Piuttosto spesso sarete deliberatamente guidati in situazioni con errori. Lo scopo, naturalmente, è di introdurvi gradualmente nell'ambiente della programmazione. Questo è un rapporto attivo tra voi e il vostro computer che non dovrebbe essere disprezzato. Il tempo speso in questa parte pratica vi farà risparmiare molto più tempo successivamente.
In ogni capitolo, dopo gli esercizi pratici, viene svolta una discussione completa che copre tutti gli argomenti del capitolo una seconda volta. Poiché avrete già visto i concetti e le idee in azione sul computer, lo studio di questo materiale sarà più facile e fruttuoso.
In ogni capitolo sono inclusi dei programmi tipo. Questi vengono discussi nei particolari per evidenziare come un'insieme di istruzioni dia origine ad un programma in Basic. Naturalmente lo scopo finale di tutti i capitoli è di imparare come si scrivono e si eseguono dei programmi in Basic sull'home computer della Texas Instruments. Siate quindi sicuri di rivolgere un tempo sufficiente allo studio e alla comprensione di tutti gli esercizi.
Ogni capitolo, a partire dal capitolo 4, contiene una collezione di problemi. Dovrete cercare di risolvere un discreto numero di questi problemi, in modo da essere sicuri di poter scrivere programmi al livello appropriato a quel capitolo.

Le soluzioni dei problemi dispari sono date alla fine del libro. Infine, ogni capitolo (tranne il primo), contiene degli esercizi. Il loro scopo è di permettervi di rivedere quello che avrete appreso ed evidenziare le parti che richiedono ulteriore studio. Le risposte agli esercizi si trovano in una sezione a parte alla fine del libro.

Fare conoscenza con l'home computer

Poiché il vostro primo contatto con il computer vi può sembrare complicato, procederemo molto lentamente. È sicuro che dopo aver fatto un po' di pratica, il modo di operare vi sembrerà molto naturale e non vi creerà problemi. Inizialmente però preparatevi a qualche difficoltà. Se necessario, non esitate a rivedere quanto già studiato.

ARGOMENTI

In questo capitolo vogliamo familiarizzare con il computer ed iniziare ad imparare come esso opera. Fino al prossimo capitolo non si farà niente di programmazione in Basic. Tuttavia, imparare come agisce la tastiera, come si inseriscono e si modificano le informazioni, è fondamentale per tutto quello che segue. È molto semplice possedere una discreta padronanza di questo materiale, ma accertatevi di avere una visione veramente chiara del contenuto del capitolo.

Collegamento del computer al video TV

L'home computer TI usa un video a colori come principale mezzo di output. Per informazioni dettagliate riguardo a questo collegamento consultate il manuale.

Modo immediato

La maniera più semplice di usare il computer è nel modo immediato. Non si tratta di eseguire un programma, semplicemente il computer esegue le istruzioni non appena vengono introdotte. A tempo debito vedremo come ottenere di più, ma per ora le semplici operazioni effettuate nel modo immediato sono una buona introduzione all'operare del computer.

Editing sul video

Raramente si riescono a dare informazioni al computer senza fare errori. Noi dobbiamo essere in grado di correggere facilmente le informazioni date. Una completa padronanza di questa caratteristica vi farà risparmiare parecchio tempo.

PRATICA SUL CALCOLATORE

Prima di iniziare a lavorare sul computer dobbiamo fissare parecchi punti importanti. Su una macchina da scrivere, la lettera L minuscola è usata per il numero 1. Sul computer invece viene usato un tasto differente. Il numero 1 si trova con gli altri numeri sulla parte superiore della tastiera. Uno degli errori più frequenti fatti dai principianti è di scrivere L al posto di 1. Oltre a questo non usate la O maiuscola al posto del numero 0. Come l'1, anche lo 0 si trova assieme agli altri tasti numerici.

Ora siamo pronti ad iniziare a lavorare. Sedetevi di fronte al computer e mettetevi comodi. Partiamo!

1. Primo, accendete la TV. Poi accendete l'interruttore localizzato in basso a destra nella consolle. Dopo qualche momento vedrete un messaggio che inizia con PRESS ANY KEY TO BEGIN (premi un tasto qualsiasi per cominciare). Seguite le istruzioni e premete un tasto. Questo fa apparire la lista delle possibili scelte. Poiché la prima scelta è TI BASIC e poiché questo libro è dedicato esclusivamente al Basic a questo punto va sempre scelta l'opzione 1. Ora scrivete

   ```
   PRINT 1+4
   ```

   e fermatevi. È successo qualcosa?

Premete ENTER e annotate qui sotto quello che il computer ha fatto.

---

2. Ora sapete come fare un'addizione al computer. Vediamo qualcos'altro. Scrivete

```
PRINT 20.1+54
```

e premete ENTER. Che cosa è successo?

---

3. Scrivete

```
PRINT 2+4-3
```

e premete ENTER. Annotate la risposta qui sotto.

---

4. Va bene, il "+" ed il "–" sono abbastanza semplici. Scrivete la seguente espressione

```
PRINT 12/2
```

e premete ENTER. Che cosa è successo?

Che operazione aritmetica indicherà il simbolo "/"?

---

5. Se scrivendo qualcosa fate un errore, esso può essere cancellato muovendo il cursore all'indietro fino all'errore premendo il tasto SHIFT-S (nel TI 99/4A premere FCTN-S). Ogni volta che si preme tale tasto il cursore si muoverà di un posto a sinistra. Appena raggiunto l'errore riscrivete la riga in modo corretto. Quando si preme il tasto ENTER il computer può rispondere con INCORRECT STATEMENT. Se questo succede cercate di scoprire qual è l'errore e riscrivete la linea.

6. Il vostro video TV dovrebbe essere ora abbastanza pieno. Scrivete CALL CLEAR e premete ENTER. Che cosa succede?

___

7. Ora che sapete come fare, potete cancellare il video ogni volta che volete. Se lo schermo è pieno e si scrivono altre linee, quelle in cima scompariranno verso l'alto. Proseguiamo nell'esplorazione del funzionamento del modo immediato. Scrivete

```
PRINT 2*50
```

e premete ENTER. Che cosa è accaduto?

___

Che operazione aritmetica indica il simbolo "*"?

___

8. Scrivete la seguente espressione ma non premete ENTER quando avete finito.

```
PRINT (2+3)*4-1
```

Che cosa succederà quando premerete ENTER?

___

Premete ENTER ed annotate sotto quello che è successo.

___

9. Passiamo ora ad un nuovo problema. Scrivete

```
PRINT "(2+3)*4-1"
```

e premete ENTER. Che cosa ha fatto il computer?

___

10. Che cosa accadrà se scrivete

```
PRINT "CANE CATTIVO"
```

e premete ENTER?

Provate e vedete se avete detto giusto.

11. Consideriamo adesso un argomento differente. Cancellate il video. Se avete dimenticato come si fa riguardate il passo 6. Scrivete la seguente riga e quando avete finito premete il tasto ENTER.

```
GRADO = 95
```

Ora scrivete

```
PRINT GRADO
```

e poi premete ENTER. Che cosa è successo?

---

12. Proseguiamo con questa nuova idea. Fermatevi un momento per esaminare queste righe

```
LUNG = 10
LARG = 6
ALT = 4
VOL = LUNG*LARG*ALT
PRINT VOL
```

Cosa pensate che faccia il computer eseguendo queste linee?

Ora scrivete queste linee ricordando di premere ENTER al termine di ognuna. Che cosa è successo?

---

13. Studiate brevemente le seguenti linee.

```
LUNGHEZZA = 12
LARGHEZZA = 9
METRIQUAD = LUNGHEZZA*LARGHEZZA
PRINT "METRI QUADRATI",METRIQUAD
```

Cosa farà il computer con queste istruzioni?

Pulite lo schermo e digitate le istruzioni. Ricordate di premere ENTER al termine di ogni linea.
Cosa fa il computer?

---

14. Abbiamo considerato un esempio dell'istruzione CALL in CALL CLEAR che pulisce lo schermo TV. Ora guardate alcune altre istruzioni CALL che possono essere usate. Per prima cosa pulite lo schermo. Ora scrivete

```
CALL HCHAR(12,1,88,32)
```

e premete ENTER. Cosa succede?

---

15. Benissimo, ora pulite lo schermo e scrivete quanto segue

```
CALL HCHAR(12,16,65,32)
```

Premete il tasto ENTER e annotate quanto appare sullo schermo.

---

16. Pulite lo schermo e provate ancora:

```
CALL VCHAR(1,1,90,768)
```

Questa volta osservate attentamente quello che appare quando premete il tasto ENTER. Che cosa è successo?

---

17. Bene, passiamo ad un altro argomento. Pulite lo schermo e scrivete le seguenti linee. Ricordate di lasciare gli spazi dove sono indicati. Al termine di ogni riga premete il tasto ENTER. Assicuratevi di regolare il volume del vostro apparecchio televisivo.

```
TEMPO = 1000
NOTA = 440
CALL SOUND(TEMPO,NOTA,0)
```

Dovreste aver sentito una nota. Vero?

Provate ancora un po' questo comando. In particolare cambiate il valore della variabile TEMPO da 100 a 3000.
Provate altri valori per la variabile NOTA (restando nell'intervallo da 440 a 880). Dopo pochi tentativi dovreste essere in grado di prevedere cosa produce il comando CALL SOUND.

18. Questo conclude per ora la pratica al calcolatore. Scrivete BYE e premete ENTER. Spegnete il computer e passate alla discussione.

DISCUSSIONE

Torneremo ora agli argomenti sui quali avete appena lavorato sul computer. Con questa esperienza sarete avvantaggiati nel capire la discussione.

Come si accende e si spegne il computer

È molto facile accendere e spegnere il computer! Come avete già visto questo si fa con un interruttore posto nella consolle in basso a destra. Una volta acceso l'apparecchio TV, quando si accende il computer vi apparirà il messaggio:

```
       TEXAS INSTRUMENTS
         HOME COMPUTER

READY-PRESS ANY KEY TO BEGIN
```

Se premete un tasto qualsiasi il computer presenta le seguenti possibili alternative (nel TI 99/4A tale scritta è diversa.)

```
1 FOR TI BASIC
2 FOR EQUATION CALCULATOR
3 (opzionale)
```

Se è inserito uno dei moduli di comando forniti dalla Texas Instruments, questo viene specificato al numero 3. Se non è inserito alcun modulo sss, la terza riga della lista sarà assente. Noi ci riferiremo sempre alla prima opzione TI BASIC.
Un punto importante: se in un momento qualsiasi perdete il controllo della situazione c'è un modo semplice per riottenerlo. Premete semplicemente il tasto SHIFT e la lettera Q. Questo vi riporterà al punto di partenza come dopo aver acceso il computer. D'altra parte, premere SHIFT-Q è equivalente a scrivere BYE. In ogni caso, sia scrivendo BYE che pre-

mendo SHIFT-Q, tutti i guai precedenti saranno annullati e il computer sarà nuovamente pronto a lavorare. Questo rimedio non è privo però di inconvenienti, dato che tutte le informazioni contenute nella memoria saranno perse nel momento in cui premete SHIFT-Q. Tuttavia questo è un modo sicuro per riprendere in mano la situazione.
Naturalmente se doveste premere inavvertitamente lo stesso tasto mentre introducete dati nel computer uscireste immediatamente dal Basic e tornereste alla situazione iniziale. Per questo dovrete fare attenzione.

Modo immediato

Nella parte pratica sul calcolatore avete imparato come fare delle semplici operazioni aritmetiche usando il computer come una calcolatrice. Questo è anche conosciuto come modo "esecuzione immediata". Come vedremo nel prossimo capitolo, il Basic memorizza istruzioni e comandi in una serie di linee numerate, dopo di che gli si potrà ordinare di eseguirle tutte sequenzialmente. Se però le istruzioni vengono inserite senza un numero di linea, il computer capirà che volete una risposta immediata e farà quello che gli avrete chiesto, se possibile.
Quando scrivete un'istruzione, non appare nulla finché non premete ENTER. Quando premete il tasto ENTER dite al computer che avete terminato e che deve iniziare ad elaborare le informazioni. Ricordate: quando avete terminato di scrivere e volete introdurre i dati nel computer, premete il tasto ENTER.
Ci sono pochi casi nei quali il computer risponde ad un singolo tasto senza che venga premuto ENTER. Un esempio di ciò è l'istruzione PRESS ANY KEY TO BEGIN che appare al momento dell'accensione. Tuttavia questi casi costituiscono l'eccezione piuttosto che la regola. Abbiamo visto che la somma e la sottrazione sono indicate rispettivamente con "+" e "–", il che probabilmente non è stata una gran sopresa! La moltiplicazione e la divisione si indicano rispettivamente con "*" e con "/". Per raggruppare le operazioni in un qualunque modo si possono usare le parentesi. Sono possibili parecchie altre operazioni ma saranno discusse nei capitoli successivi.
Se scrivete

```
PRINT 5*3.2+6.3
```

e premete ENTER il computer eseguirà i calcoli aritmetici e scriverà il risultato. Se scrivete

```
PRINT "ABCDEFG"
```

e premete ENTER il computer stamperà la sequenza di caratteri posti tra virgolette, in questo caso le lettere ABCDEFG. Una tale sequenza è chiamata *stringa di caratteri*, ed è un importante concetto sul quale ritorneremo lungo tutto il libro.
Il computer memorizza un certo numero di informazioni nel modo immediato. Ad esempio

```
A = 2
B = 3
PRINT A+B
```

farà apparire sullo schermo il numero 5. C'è un punto molto importante collegato a questo concetto. Se scrivete

```
PRINT TAX
```

e premete ENTER apparirà il numero 0. Finché non verrà assegnato alcun valore a TAX, il computer attribuirà a questa variabile il valore 0 che verrà visualizzato.
Il computer è molto elastico nell'accettare i nomi delle variabili numeriche, sia nel modo immediato che nei programmi in Basic. Potete usare nomi "lunghi" come LARGHEZZA o PREZZO allo stesso modo di quelli "corti" come L o P. Tuttavia nell'usare nomi lunghi bisogna fare attenzione a certe cose. I nomi non devono contenere spazi. Gli spazi hanno un significato sia nel modo immediato che nei programmi Basic. Certe parole non possono essere usate come nomi di variabili perché sono riservate al computer stesso. Consultate il manuale per la lista delle parole riservate.
Questa breve introduzione sui nomi delle variabili è sufficiente per la nostra discussione sul modo immediato. Torneremo ad approfondire questo concetto in seguito. Durante l'attività pratica avete incontrato diversi esempi dell'istruzioni CALL. Tale istruzione viene usata nei programmi Basic per migliorarne l'efficienza. Poiché siamo appena agli inizi dello studio del Basic, potremo rinviare un esame appronfondito della funzione CALL al capitolo 11. L'unico motivo per cui abbiamo introdotto questo argomento è che l'istruzione CALL può essere usata come comando immediato. Comunque dobbiamo discutere un aspetto importante connesso al comando CALL CLEAR. Come vedrete nel capitolo 7, i caratteri che appaiono sullo schermo possiedono un particolare numero di codice. Ad esempio, il carattere numero 32 è uno spazio. CALL CLEAR riempie tutto il monitor TV con il carattere numero 32. Naturalmente così si cancella tutto quello che c'è sul video esattamente come vogliamo.
È possibile ridefinire in carattere numero 32 con qualche carattere differente. In questo caso CALL CLEAR riempirà lo schermo con questo nuovo carattere. Sarete a dir poco sorpresi di vedere lo schermo riempito

con questi strani caratteri usando il CALL CLEAR invece di vederlo pulito come vi aspettavate e penserete, a torto, che qualcosa non vada nel vostro computer. Se doveste avere questa esperienza sapete ora a cosa è dovuta.

Editing su video

Il TI home computer possiede comandi per l'editing di linea che permettono di effettuare correzioni. Questi sono molto utili quando vengono usati per modificare programmi Basic. Tuttavia, poiché alcuni di questi comandi possono essere usati nel modo immediato, studieremo il processo in dettaglio.
Limiteremo la nostra discussione ai cambiamenti che si possono fare in una linea prima di premere il tasto ENTER. Per prima cosa il cursore può essere spostato avanti e indietro con i tasti SHIFT-S e SHIFT-D (nel TI 99/4A, rispettivamente FCTN-S e FCTN-D). Le frecce su questi tasti vi aiutano a ricordare la loro funzione. Il cursore può muoversi sopra i caratteri della linea senza cambiarli. Se viene premuto un carattere esso prende il posto di quello presente sotto il cursore. Si possono anche inserire o cancellare caratteri in una riga. Se premete il tasto SHIFT-G (per il TI 99/4A FCTN-2) e scrivete delle lettere, queste vengono inserite iniziando dalla posizione del cursore. I vecchi caratteri slittano verso destra mentre vengono inseriti i nuovi. Se si preme il tasto SHIFT-F (per il TI 99/4A FCTN-1) viene cancellato il carattere sotto il cursore e tutto ciò che compare a destra si sposta a sinistra di una posizione. Premendo SHIFT-F diverse volte, si possono cancellare da una linea quanti caratteri si vogliono.
Questi semplici comandi di editing possono essere usati in modo immediato per effettuare cambiamenti o correzioni.
Ricordate che si possono usare solo se non avete ancora premuto ENTER. Nel prossimo capitolo vedremo in modo più approfondito le varie possibilità offerte dai comandi di editing quando vengono usati nei programmi Basic.

ESERCIZI

Considerate il testo che segue e controllate quanto avete imparato degli argomenti del capitolo 2. Le risposte agli esercizi sono date alla fine del libro.

1. Quando avete finito di scrivere una linea, come fate a dire al compu-

ter che avete terminato?

---

2. Nel caso perdiate il controllo del computer, come potete riacquistarlo?

---

3. Che simbolo si usa per indicare la moltiplicazione?

---

4. Come si fa a cancellare lo schermo?

---

5. Che operazione indica il simbolo "/"?

---

6. Che cosa succederà se scrivete

```
PRINT 3*4/6
```

e poi premete ENTER?

---

7. Cosa accadrà se scrivete

```
PRINT "25/5+2"
```

e poi premete ENTER?

---

8. Supponete di aver scritto PRING 2+3*4 e prima di premere ENTER vi accorgete che c'è una G invece di una T nella parola PRINT. Descrivete esattamente come correggereste l'errore.

CAPITOLO TRE

Introduzione al Basic

Siamo ora pronti ad iniziare lo studio del Basic. In questo capitolo vedremo come scrivere ed eseguire alcuni programmi molto semplici.

ARGOMENTI

Gli argomenti sono semplici ma importanti, dato che rappresentano la vostra prima introduzione al Basic. Eccone una lista.

Le caratteristiche dei programmi in Basic

Tutti i programmi in Basic hanno delle caratteristiche comuni. Consideremo alcuni semplici programmi per capire quali sono.

I comandi di sistema

I comandi di sistema sono usati per dire al computer di fare qualcosa a un programma o con un programma. Vedremo i seguenti comandi di sistema: LIST, RUN, STOP, e NEW che vengono usati per controllare un programma.

La scrittura ed il controllo dei programmi

Questo argomento include il precedente. Il nostro scopo è di rendervi sicuri quando scrivete o controllate dei programmi. All'inizio tutti i programmi che incontreremo saranno corti e facili da trattare.

I nomi delle variabili in Basic

Poiché soltanto certe combinazioni di caratteri possono essere usate per dare nomi (per rappresentare) numeri o stringhe, dobbiamo conoscerle bene. Fortunatamente il computer in questo caso ha regole molto elastiche.

PRATICA SUL CALCOLATORE

Negli esercizi al calcolatore che seguono avrete a che fare con dei programmi. Se vedete un < ENTER > nelle istruzioni, premete ENTER. Vi ricorderete dalle esperienze del capitolo 2 che la pressione del tasto ENTER indica al computer che avete finito di scrivere. Passiamo ora agli esercizi.

1. Accendete il vostro computer e selezionate il Basic. Scrivete:

```
100 LET A=1        <ENTER>
```

   Questa è la prima linea di programma in Basic. Notate il ">" alla sinistra dello schermo dove andrà la prossima linea.

2. Inserite ora il resto del programma che segue qui sotto.

```
110 LET B=8       <ENTER>
120 LET C=A+B       <ENTER>
130 PRINT C       <ENTER>
140 END       <ENTER>
```

   Se mentre scrivete il programma fate degli errori correggeteli usando il cursore e i metodi visti nel capitolo 2.

3. Pulite lo schermo usando il comando CALL CLEAR. Che cosa è successo al programma che avete appena inserito?

   ____

4. Fortunatamente non è andato tutto perso. Il computer ha ricordato quello che avete scritto anche se lo schermo è vuoto. Scrivete LIST e premete ENTER. Che cosa è successo?

____

5. Sul video potete vedere il programma appena inserito. Per il momento, ignorate i numeri all'inizio di ogni linea. Leggete soltanto le linee del programma e cercate di capirne il significato. Se si dicesse al computer di eseguire le istruzioni, cosa pensate che accadrebbe?

____

Scrivete RUN e premete ENTER. Cosa è successo?

____

6. Va bene. Ora scrivete

```
110 LET B=5        <ENTER>
```

Pulite lo schermo, scrivete LIST e premete ENTER. Che cosa è successo alla linea 110 del programma?

____

7. Se fate eseguire al computer questo programma, che cosa pensate che accadrà?

____

Ora osservate come cambia il colore dello schermo quando il programma è stato eseguito. Scrivete RUN e premete ENTER. Prendete nota di quanto è successo. Avevate detto giusto?

____

8. Ora scrivete

```
140     <ENTER>
```

Cancellate lo schermo e fate una lista del programma usando il comando LIST. Che cosa è successo alla linea 140?

____

Quindi che cosa si deve fare per eliminare una riga?

---

9. Eseguite il programma. Che cosa è accaduto?

Vi sembra che l'istruzione END che prima era nella linea 140 sia necessaria al computer?

---

10. Facciamo qualche altro esperimento. Capita spesso di voler cancellare il programma dalla memoria del computer. Questo si fa con il comando NEW. Scrivete NEW e premete ENTER. Vi sembra che sia successo qualcosa?

Scrivete LIST e premete ENTER per vedere quello che il computer ha in memoria. C'è qualcosa?

---

11. Abbiamo visto come eliminare un programma dalla memoria, ma ora non lo abbiamo più. Per riavere il nostro programma dobbiamo riscriverlo. Inserite il programma che segue

```
100 LET A=1         <ENTER>
110 LET B=8         <ENTER>
120 LET C=A+B        <ENTER>
130 PRINT C         <ENTER>
140 END         <ENTER>
```

Controllate tutte le linee per essere sicuri che siano state scritte correttamente. Se una linea va modificata riscrivetela. Se dovete riscrivere delle linee, pulite lo schermo con CALL CLEAR e listate nuovamente il programma scrivendo LIST.

12. Ora scrivete

```
125 LET D=B-A        <ENTER>
135 PRINT D        <ENTER>
```

Cancellate lo schermo e listate il programma. Che cosa è successo?

---

13. Prendetevi qualche minuto per studiare il programma. Cosa accadrà se lo eseguite?

    ____

    Scrivete RUN, premete ENTER ed annotate il risultato.

    ____

14. Nel programma originale, i numeri delle linee non erano consecutivi (come 100, 101, 102, 103, ecc.), ma avevano dei salti (p. es. 100, 110, 120, 130 e 140). Potete ora pensare ad una ragione per cui questo viene fatto? (Suggerimento: rivedete il passo 12.)

    ____

15. Come si fa ad inserire delle linee in un programma Basic? (Suggerimento: rivedete i passi 12 e 14.)

    ____

16. Cancellate il programma dalla memoria scrivendo NEW e premendo ENTER. Inserite il seguente programma.

```
100 INPUT BIANCO        <ENTER>
110 LET ROSSO=BIANCO+2       <ENTER>
120 PRINT ROSSO       <ENTER>
130 GOTO 100       <ENTER>
140 END        <ENTER>
```

17. Questo nuovo programma ha parecchie caratteristiche che finora non avete viste. Studiatelo attentamente e pensate che cosa accadrebbe se lo facessimo funzionare. Che significato ha il GOTO 100 nella linea 130?

    ____

18. Ora eseguite il programma e segnate che cosa ha fatto il computer.

    ____

    Scrivete in numero 6 e premete ENTER. Che cosa è successo?

    ____

19. Scrivete il numero 10 e date ENTER. Che cosa è accaduto?

    ____

20. Quale pensate sia la linea che genera il punto interrogativo?

____

Descrivete con parole vostre cosa fa il programma. Se necessario provatelo ancora un po' per essere sicuri di aver capito bene.

____

21. Vogliamo ora uscire dal programma. Premete i tasti SHIFT e C contemporaneamente. Da adesso useremo l'annotazione "SHIFT-C" (FCTN-4). Che cosa è successo?

____

22. Liberate la memoria del computer ed inserite il seguente programma.

```
100 LET A=1        <ENTER>
110 PRINT A        <ENTER>
120 LET A=A+1        <ENTER>
130 GOTO 110        <ENTER>
140 END        <ENTER>
```

23. Eseguite il programma e segnate quello che è successo.

____

Quando siete stanchi di guardare il video premete il tasto SHIFT-C. Che cosa è accaduto?

____

24. Provate ancora una volta. Eseguite il programma, e quando sono stati stampati dei numeri, interrompetelo. Come si ferma un programma che è in esecuzione?

____

25. Cancellate lo schermo e listate il programma in memoria. Scrivete le due linee che seguono. Osservate l'assenza di spazi nella prima linea e gli spazi in più nella seconda.

```
100LETA=1        <ENTER>
1 2 0 L E T A = A + 1        <ENTER>
```

Ora cancellate il video e listate il programma. Naturalmente gli spa-

zi sono importanti nelle istruzioni Basic. Ritorneremo su questo punto più avanti.

---

26. Proviamo un programma con qualche nuova caratteristica. Eliminate il contenuto della memoria scrivendo NEW e premendo ENTER. Inserite il seguente programma.

```
100 PRINT "SCRIVI UN NUMERO"      <ENTER>
110 INPUT PRIMO      <ENTER>
120 PRINT "ANCORA UNO"      <ENTER>
130 INPUT SECONDO      <ENTER>
140 LET SOMMA=PRIMO+SECONDO      <ENTER>
150 PRINT "LA LORO SOMMA E'"      <ENTER>
160 PRINT SOMMA      <ENTER>
170 END      <ENTER>
```

27. Studiate il programma per qualche momento. Poi fatelo funzionare. Che cosa è successo?

Scrivete il numero 12, premete ENTER ed annotate il risultato.

---

28. Va bene; ora scrivete il numero 13, premete ENTER e segnate quello che è successo.

---

29. Questo semplice programma illustra la possibilità che un programma stampi sia dei numeri che dei messaggi.

30. Passiamo ora ad un argomento differente. Premete CLR per cancellare il video. Scrivete NEW e premete ENTER per liberare la memoria. Poi inserite il seguente programma:

```
100 LET A=1      <ENTER>
110 LET A$="CASA"      <ENTER>
120 PRINT A      <ENTER>
130 PRINT "A"      <ENTER>
140 PRINT A$      <ENTER>
150 PRINT "A$"      <ENTER>
160 END      <ENTER>
```

31. Questo programma contiene qualcosa di nuovo. Guardate l'A$ nella linea 110. Notate che è posta uguale ad una parola che è racchiusa tra virgolette. Il resto del programma riguarda vari tipi di stampa

di A e di A$. Eseguitelo ed annotate i risultati.

---

32. Studiate i risultati attentamente ed identificate cosa è stato stampato in risposta ad ogni istruzione PRINT. Per il momento fate soltanto un confronto. Più avanti questo sarà esaminato nei dettagli. Inserite la seguente linea

```
155 PRINT B        <ENTER>
```

33. Cancellate il video e listate il programma con il comando LIST. Notate che l'unico posto in cui compare B è nella linea 155, nell'istruzione PRINT. Che cosa pensate che accadrà se eseguiamo il programma?

Bene, ora fate funzionare il programma e segnate che cosa è successo.

---

34. Come avete visto, anche se il valore di B non è stato definito nel programma, il computer assegna a B il valore 0. Questo è un fatto importante, da considerare mentre si scrivono dei programmi. Ritorneremo su questo più avanti.

35. Ora vedremo una possibilità che potrà esservi di molto aiuto quando scriverete programmi. Pulite lo schermo e listate il programma. Fate attenzione ai numeri di linea. Ora scrivete RES 1000,10 e premete ENTER. Listate nuovamente il programma. Che cosa è successo?

---

36. Provate un'altra volta. Scrivete RES 200,5 e premete ENTER. Listate il programma. Cosa è accaduto?

Avete capito la funzione del comando RES?

---

37. Questo conclude la parte pratica del capitolo. Scrivete BYE e premete ENTER. Spegnete il computer e passate al prossimo paragrafo.

DISCUSSIONE

Ora che avete terminato la parte pratica sul calcolatore avete visto alcune caratteristiche del Basic in azione, possiamo riassumere quello che è stato fatto.

Correzione degli errori

Poiché molti di noi fanno degli errori mentre scrivono, è necessario poterli correggere. Supponiamo di fare un errore mentre stiamo scrivendo una linea. Il modo di correggerlo dipende da dove si trova l'errore e dal fatto che l'ENTER sia stato premuto o no. Prima di premere il tasto ENTER è possibile spostare il cursore avanti e indietro per effettuare correzioni. Il tasto SHIFT-S (FCTN-S) sposta il cursore a sinistra e il tasto SHIFT-D lo sposta a destra. Nel capitolo 2 avete già visto come inserire dei caratteri dopo aver premuto il tasto SHIFT-G (FCTN-2). È anche possibile cancellare caratteri usando il tasto SHIFT-F (FCTN-1). È disponibile una striscia da inserire nella parte superiore della tastiera la quale identifica le funzioni dei tasti usati per l'editing. Quando sono state effettuate tutte le correzioni va premuto il tasto ENTER. Notate come non sia necessario che il cursore venga a trovarsi all'estrema destra quando si preme ENTER. Quando premete ENTER la linea che avete scritto (compresi eventuali errori non corretti) viene analizzata dal computer. Alcuni errori vengono individuati a questo livello, nel qual caso il computer scriverà *INCORRECT STATEMENT. Certi errori vengono evidenziati durante l'esecuzione del programma. Se il computer ne scopre uno in questa fase, stamperà un messaggio d'errore assieme ad un numero di linea. Supponiamo che il computer abbia trovato un errore alla linea 350. Se scrivete EDIT 350, apparirà tale linea sullo schermo. Ora la linea può essere corretta usando i tasti di editing. Abbiamo a disposizione un'altra alternativa per effettuare cambiamenti in una linea. Se premiamo il tasto ENTER, le correzioni diventano effettive ma il computer esce dal modo edit. Invece se non premiamo tale tasto il computer rimane nel modo edit. Se ci sono altre correzioni da fare nelle linee vicine basta premere i tasti che portano impresse le frecce verso l'alto o verso il basso, a seconda della necessità, per fare apparire le linee desiderate sullo schermo. Quando si segue tale metodo, il computer memorizza i cambiamenti effettuati in una linea nel momento in cui una nuova linea viene posta sul-

lo schermo. In questo modo il computer rimane nel modo edit. Quando sono state fatte tutte le correzioni, premete ENTER per porre in memoria le ultime modifiche ed abbandonare il modo edit.

Caratteristiche dei programmi in Basic

Sono state viste parecchie particolarità importanti dei programmi in Basic. Per avere un programma da usare nella discussione, torneremo a quello originale usato nella parte pratica:

```
100 LET A=1
110 LET B=8
120 LET C=A+B
130 PRINT C
140 END
```

Ogni programma in Basic consiste in un gruppo di linee chiamate *istruzioni*. Ogni istruzione deve avere un numero di linea. Nel programma sopra, ci sono tre tipi di istruzioni Basic: assegnazione (identificata dal segno "="), PRINT e END. Le prime due verranno trattate a fondo nel prossimo capitolo. Per il momento, l'uso di queste istruzioni nel programma è chiaro. L'istruzione END, tuttavia, ha un significato particolare. Come avete visto negli esercizi sul calcolatore, il computer non richiede la presenza dell'END. Comunque, è buona norma mettere l'istruzione END nei programmi. La presenza di questa istruzione è un chiaro segno che il programma è finito.

Generalmente, in un programma i numeri di linea non sono consecutivi (come 100, 101, 102, ecc.). Il motivo è che successivamente potremmo voler inserire delle altre istruzioni, se scopriamo degli errori o se vogliamo modificare il programma. Se le linee fossero numerate consecutivamente, per poter fare dei cambiamenti bisognerebbe riscrivere l'intero programma. Saltando dei numeri, l'inserimento di istruzioni si effettua semplicemente utilizzando dei numeri di linea che non compaiono già nel programma.

Molto spesso dopo aver effettuato dei cambiamenti in un programma si desidera mettere in ordine i numeri di linea. Questo non ha nessuna influenza sull'esecuzione del programma, ma semplicemente rende il listato più piacevole da vedere. Il comando *resequence* è usato per rinumerare le linee in un programma. Scrivendo RES M,N dove M ed N sono numeri, il programma viene numerato cominciando da M e con spaziatura N. Ad esempio RES 1000,100 farà in modo che il programma venga numerato a partire dal numero 1000. Il secondo sarà 1100, il terzo 1200 e così via. Più avanti scopriremo che i programmi Basic possono effettua-

re salti ad ogni linea del programma. Il comando resequence tiene conto di questi salti quando vengono cambiati i numeri di linea corrispondenti. Al computer non importa in che ordine vengono inserite le linee di un programma. Se per esempio, scriviamo

```
140 END
120 LET C=A+B
110 LET B=8
130 PRINT C
100 LET A=1
```

e listiamo il programma, il computer ordinerà le istruzioni e le farà apparire in successione numerica. Allo stesso modo, se ordiniamo al computer l'esecuzione del programma, prima di iniziare effettuerà l'ordinamento delle istruzioni. Potete togliere un'istruzione Basic da un programma scrivendo il numero di linea e premendo ENTER. Si possono modificare delle istruzioni riscrivendo le linee interessate e premendo ENTER alla fine di ogni linea, oppure usando l'editor. Come indicato sopra si possono aggiungere istruzioni utilizzando numeri di linea non ancora usati nel programma. Quindi le istruzioni possono essere aggiunte, eliminate o modificate a piacere. La capacità di cambiare facilmente i programmi è una delle caratteristiche potenti del Basic.
Se desiderate, potete chiedere al computer di fornire automaticamente i numeri di linea mentre scrivete un programma. Se scrivete NUM 1000,10 il computer fornisce il numero di linea 1000 e aspetta che voi scriviate la linea. Quando viene premuto ENTER, viene dato il numero 1010 per una linea successiva e il computer aspetta una nuova istruzione. In generale NUM M,N serve a dare automaticamente i numeri di linea cominciando da M e con incremento N. Se avete terminato di scrivere un programma ed è già pronto il numero per una linea successiva, premete il tasto ENTER per uscire dal modo *number*.
Un ultimo commento sul Basic riguarda gli spazi vuoti nelle istruzioni. Il computer accetta spazi nelle istruzioni Basic solo in determinati punti. Il vostro buon senso vi farà da guida. Non ponete spazi nei numeri di linea, nei nomi di variabili, nelle parole che indicano un comando o nei numeri. Ad esempio, la seguente istruzione Basic è scorretta.

```
1 06L ETX=1.03 58
```

Ci sono spazi nel numero di linea, nella parola LET e nel valore assegnato alla X. Con spazi posti in questo modo risulta difficile leggere! Generalmente, se inserirete gli spazi in modo da rendere più facile la lettura dell'istruzione, non avrete nessuna difficoltà. La seguente riga illustra il modo corretto di scrivere.

```
106 LET X=1.0358
```

Non dovete preoccuparvi troppo. Se commettete un errore il computer vi avvertirà. Dopo alcune ore di programmazione, l'uso corretto delle spaziature nelle istruzioni Basic diventerà una cosa naturale.

Dare dei comandi al computer

Dobbiamo fare una sottile distinzione tra le istruzioni di un programma e i comandi di sistema. Questi ultimi dicono al computer di fare qualcosa con un programma. Ne abbiamo visti parecchi nella parte pratica; rivedremo brevemente l'uso di ognuno.
Capita piuttosto spesso di volere una lista del programma che c'è in memoria, per esempio per il fatto che alcune modifiche del programma hanno prodotto una cosa poco chiara sul video. Il modo di risolvere il problema è di far listare il programma che sta in memoria. Questo si ottiene con il comando LIST. Se scrivete LIST e poi premete ENTER, il computer farà una lista del programma sullo schermo. Solitamente, prima cancellerete il video, in modo da ottenere una copia pulita del programma. Se il programma ha più di 24 linee e viene visualizzato con il comando LIST, si vedranno solo le ultime 24. Qualsiasi cosa venga prima delle 24 linee finali verrà persa uscendo dalla parte alta del video. Modificando il comando LIST è però possibile vedere una qualsiasi parte di un programma indipendentemente dalla lunghezza di quest'ultimo. Se ad esempio scriviamo LIST 300–400, il computer listerà le istruzioni del programma da 300 a 400 incluse. Oppure LIST–200 causerà la stampa delle istruzioni dall'inizio fino alla linea 200. LIST 300 mostrerà solo la linea 300. Infine LIST 400– dirà al computer di fare una lista delle linee da 400 fino alla fine del programma.
Un programma in Basic è semplicemente un insieme di istruzioni che devono essere eseguite dal computer. È però necessario dire al computer di iniziare questo procedimento. Questo si fa con il comando RUN. Quando il comando RUN è ricevuto, il computer va all'istruzione con il numero di linea più basso, la esegue, passa alla successiva e continua ad eseguire istruzioni in ordine numerico, a meno che il programma non indichi di seguire un ordine diverso. Ricordate quindi: il computer inizia ad eseguire un programma se scrivete RUN e premete ENTER.
Una caratteristica molto apprezzabile del TI home computer è che i colori del video cambiano durante l'esecuzione del programma. Mentre scrivete le lettere sono nere su sfondo blu chiaro. Invece quando il programma è in esecuzione lo sfondo cambia in verde chiaro. Pertanto il colore del video vi fornisce un semplice modo per determinare se il computer sta eseguendo un programma o no.

Supponiamo che abbiate finito di lavorare con un programma e decidiate di passare ad un altro. Potete cancellare il video, ma questo non elimina il programma corrente dalla memoria. Una parte della memoria del computer ricorda ciò che è visualizzato sul video. Una parte separata contiene il programma corrente. Con il tasto CLR segnalate al computer di cancellare qualsiasi cosa si trovi nella parte di memoria dedicata allo schermo. Il comando NEW è usato per eliminare il programma corrente dalla memoria. Avete visto lavorando sul calcolatore che il comando NEW non ha effetti su quello che compare sul video. Poiché, come ora sappiamo, la memoria è formata da due parti distinte, questo non dovrebbe creare confusione. Dovreste sempre usare il comando NEW quando non vi serve più un programma. Se il vecchio programma non è stato cancellato, quello nuovo andrà nello stesso spazio e, come risultato, il computer potrà avere in memoria parti di programmi differenti.

Inserimento e controllo dei programmi

Finora, quando dovevate inserire comandi o istruzioni di programma, trovavate scritto < ENTER > per ricordarvi di scrivere ENTER. Adesso dovreste essere abituati a farlo, perciò, d'ora in poi, non sarà più usato. Sorgono situazioni dove è necessario poter controllare un programma in esecuzione. Certamente uno dei casi più fastidiosi si ha quando un programma è chiuso in un ciclo, e continuerebbe a percorrerlo se non lo fermassimo. Lo possiamo fermare premendo il tasto SHIFT-C (FCTN-4). Fatto questo il computer interrompe l'esecuzione, ci dice BREAKPOINT AT (che linea stava eseguendo quando è stato dato lo stop).
Un problema diverso si ha quando il computer sta aspettando l'inserimento da parte nostra di un numero. Se vogliamo uscire da tale situazione, dobbiamo premere ancora SHIFT-C. Il computer esce dall'esecuzione del programma e scrive READY.

Nomi delle variabili in Basic

Passiamo adesso ad uno degli aspetti del Basic che più spesso causano problemi al principiante. Esso concerne i nomi delle variabili e la distinzione tra il nome e la quantità memorizzata sotto quel nome. Nell'istruzione

```
100 LET A=2
```

la lettera A è il nome di una variabile. Con *variabile* intendiamo che ad A possiamo assegnare differenti valori. Le istruzioni che contengono il

segno "=" sono chiamate *istruzioni di assegnazione*. Nel caso di cui sopra, alla variabile A è assegnato il valore 2. In effetti, quello che succede è che il computer ha nominato una locazione di memoria A, ed ha memorizzato 2 in quella locazione. È la stessa cosa della differenza tra una cassetta postale ed il contenuto di quella cassetta. Il numero della cassetta non cambia ma il suo contenuto può cambiare in qualsiasi momento. Solitamente, l'uso di LET in un'istruzione di assegnazione è opzionale. Questo vale anche nel caso del nostro computer. Noi useremo sempre LET nelle istruzioni di assegnazione per una ragione spiegata più avanti. Considerate la seguente istruzione

```
130 LET C=A+B
```

Questa dice al computer di prendere i numeri contenuti nelle locazioni chiamate A e B, sommarli e mettere il risultato nella locazione chiamata C. Il segno di uguale significa valutare quello che sta alla destra ed assegnare il risultato alla variabile di sinistra. Per continuare su questa strada, supponiamo di avere un'istruzione Basic del tipo

```
120 LET B=B+1
```

Se la consideriamo come un'equazione algebrica, abbiamo

$$B = B+1$$

Sottraendo B da entrambe le parti, otteniamo

$$0 = 1$$

che è indubbiamente molto strano! È quindi chiaro che il segno "=" in un'istruzione Basic non ha lo stesso significato di quello che ha in un'equazione algebrica. Invece, l'istruzione

```
120 LET B=B+1
```

indica al computer di prendere il numero contenuto nella locazione B, sommargli 1 e memorizzare il risultato nuovamente nella locazione B. L'uso del LET in queste istruzioni ci aiuta a ricordare che il segno "=" implica assegnazione e non uguaglianza.
Se memorizziamo un numero in una locazione, il contenuto precedente viene perso. Considerate le seguenti istruzioni

```
100 LET A=1
110 LET A=2*3
```

La linea 100 indica al calcolatore di fissare una locazione di memoria chiamata A e di metterci il numero 1. La linea 110 dice al computer di moltiplicare 2 per 3 e di porre il prodotto nella locazione A. Notate che l'1 memorizzato prima nella locazione A è stato perso. Arriviamo così al centro di questo argomento. La lettera A, che identifica una locazione di memoria, è chiamata una *variabile* perché il suo contenuto può essere cambiato. Il nome della locazione non cambia, ma il suo contenuto può essere cambiato a piacere.
Per essere precisi, la variabile A è chiamata *variabile numerica*. La ragione per includere "numerica" nel nome, è che c'è un altro tipo di variabile chiamata *stringa di caratteri*. Siete stati brevemente introdotti a questo concetto nella parte pratica; ora lo vedremo più nei dettagli.
Per quanto riguarda i nomi, è facile distinguere tra variabili numeriche e variabili di stringa. A, B, M e P indicano tutte variabili numeriche. A$, B$, M$ e P$ sono tutti nomi di stringhe di caratteri. Il simbolo "$" che viene aggiunto, identifica i nomi come variabili di stringa. Nell'istruzione

```
100 LET B$="GRANAIO"
```

B$ dà il nome ad una locazione di memoria nella quale è memorizzata la stringa di caratteri GRANAIO. Le virgolette indicano l'inizio e la fine della stringa, ma non ne fanno parte. Per quanto riguarda i nomi di variabili il TI home computer ha delle regole poco restrittive. Generalmente il Basic permette solo una lettera o una lettera seguita da un'unica cifra per indicare una variabile numerica, e le stesse combinazioni seguite dal simbolo $ ad indicare quelle di stringa. Il TI home computer permette di usare nomi "lunghi" sia per le variabili numeriche sia per le stringhe di caratteri. Sono consentiti i nomi fino a quindici caratteri (incluso il carattere $ nel caso di variabili di stringa). Il computer possiede una serie di parole riservate che sono usate nei comandi Basic e di sistema. Queste parole non possono essere usate come nomi di variabili. Consultate il manuale per l'elenco di tali parole riservate. Tuttavia se per errore ne usate una il computer vi avvertirà!
L'uso di lunghi nomi è molto conveniente poiché il nome della variabile può richiamare il suo significato. Ad esempio LUNGHEZZA, TEMPO, NOME$ e CHILOMETRAGGIO non hanno bisogno di ulteriori specificazioni, al contrario L, T, N$ e C dovrebbero essere spiegate. Tuttavia, se in un programma userete lunghi nomi ciò comporterà uno svantaggio che va considerato. Dovete scrivere correttamente le lettere del nome ogni volta che lo usate. Il computer tratterà CHILOMETRAGGIO e CHILOMETRAGIO come due nomi differenti.
Rivediamo ancora una volta i punti importanti. Il nome di una variabile in Basic identifica una locazione di memoria. Se la variabile è numerica,

nella locazione viene memorizzato un numero. Se la variabile è una stringa, viene memorizzata una collezione di caratteri. Il contenuto della locazione può essere modificato, ma i nomi delle locazioni rimangono gli stessi. L'istruzione di assegnazione valuta l'espressione che sta a destra del segno di uguale ed assegna il risultato alla variabile sulla sinistra. Così

```
100 LET D=A+B+C
```

dice al calcolatore di valutare l'espressione A+B+C usando i numeri memorizzati nelle locazioni che hanno per nome A, B, e C. Il risultato viene quindi posto nella locazione D.
Abbiamo soltanto scalfito l'argomento delle variabili stringa. Torneremo su questo punto parecchie volte durante il resto del libro.

ESERCIZI

Considerate il testo che segue e controllate quanto avete imparato degli argomenti del capitolo 3. Le risposte agli esercizi sono date alla fine del libro.

1. Come indicate al computer che avete finito di scrivere una linea o un comando?

   ____

2. Supponiamo che il computer stia aspettando che voi inseriate un numero ad un'istruzione INPUT in un programma. Voi invece decidete di uscire dal programma. Come fate?

   ____

3. Come fate ad interrompere un programma in esecuzione sul vostro computer?

   ____

4. Che cosa accadrà quando il seguente programma verrà eseguito?

```
100 LET A=1
110 LET B=2
120 LET C=B-A
130 PRINT C
140 END
```

   ____

5. I nomi "lunghi" di variabili quanto lunghi possono essere?

____

6. Come si elimina una linea da un programma?

____

7. Come si inserisce una linea in un programma?

____

8. Come si sostituisce una linea in un programma?

____

9. Come si ottiene una lista del programma in memoria?

____

10. Come si cancella lo schermo?

____

11. Come si cancella un programma dalla memoria?

____

12. Come dite al computer di iniziare ad eseguire un programma?

____

13. Che differenza c'è fra le variabili numeriche e le variabili stringa?

____

CAPITOLO QUATTRO

Aritmetica del computer e gestione dei programmi

ARGOMENTI

Ora che siete stati introdotti al Basic col computer, siamo pronti per affrontare dei lavori più interessanti.

Aritmetica del computer

In definitiva, tutta la matematica su un calcolatore viene fatta usando le più semplici operazioni aritmetiche. È essenziale aver compreso chiaramente il modo in cui queste operazioni vengono fatte.

Le parentesi nei calcoli

Come vedremo, tutte le espressioni matematiche da inserire nel computer devono essere scritte su una singola linea. Alcune espressioni possono essere trattate in questa maniera soltanto ponendo delle parti dell'espressione fra parentesi. Quindi, la capacità di saper usare le parentesi è indispensabile.

Numeri in notazione esponenziale

Il computer deve trattare con numeri sia molto grandi che molto piccoli. Tali numeri vengono espressi dal computer in notazione esponenziale. Dobbiamo essere in grado di riconoscere ed interpretare questa notazione, poiché il computer può scrivere dei numeri in questa forma.

Scrittura e lettura dei programmi sul registratore

Abbiamo già visto alcuni comandi di sistema. In questo capitolo ne introdurremo altri, che ci permetteranno di scrivere e leggere dei programmi sul registratore che può essere collegato al computer.

PRATICA SUL CALCOLATORE

La parte pratica di questo capitolo introduce le caratteristiche dell'aritmetica sul computer. Saranno inoltre trattati degli altri comandi di sistema per la gestione dei programmi.
Passiamo ora al lavoro sul calcolatore.

1. Accendete il computer, selezionate il TI Basic e scrivete il seguente programma:

```
100 INPUT A
110 INPUT B
120 LET C=A+B
130 PRINT C
140 END
```

   Che operazione indica il "+" alla linea 120?

   ____

2. Vediamo se è vero. Fate funzionare il programma. Quando il computer va alla linea 100, scrive un punto interrogativo, si ferma e aspetta che voi diate un valore per A. In questo caso scrivete 10. Il calcolatore passa allora alla linea 110, scrive un punto interrogativo, si ferma e aspetta un valore per B. Inserite 20. Che cosa stamperà il calcolatore?

   ____

3. Cambiate il "+" della linea 120 con un "–". Cancellate il video e

listate il programma. Eseguite il programma e al primo punto interrogativo (segnale di INPUT) date 30 per A, al secondo inserite 20 per B. Che cosa è successo?

Che operazione viene fatta con il "–" della linea 120?

---

4. Cambiate il "–" con "*", usando il cursore come descritto sopra. Cancellate lo schermo e listate il programma, e quando compaiono i segnali di INPUT (punti interrogativi), date 5 per A e 6 per B. Che cosa ha scritto il computer?

Che operazione è indicata con "*"?

---

5. Ora cambiate il "*" in "/". Eseguite il programma e quando compaiono i segnali di INPUT, inserite 45 per A e 15 per B. Che risposta ha dato il computer?

Che operazione aritmetica è indicata con "/"?

---

6. Finora abbiamo visto una singola operazione aritmetica su una linea. Vediamo un esempio in cui ce n'è più d'una. Scrivete

```
120 LET C=A+B-B/3
```

Cancellate lo schermo, listate il programma e studiatelo brevemente. Se ora lo eseguiamo ed inseriamo 2 per A e 3 per B, che cosa pensate che succederà?

Fate funzionare il programma, date i valori detti sopra e annotate quello che è successo.

---

7. Eliminate il programma dalla memoria scrivendo NEW e premendo ENTER. Poi cancellate lo schermo e scrivete

```
100 LET A=3*3
110 LET B=3^2
120 PRINT A
130 PRINT B
140 END
```

Assicuratevi di aver inserito il programma correttamente. Poi eseguitelo e segnate i risultati.

____

Confrontate i numeri stampati con le espressioni delle linee dove sono stati calcolati. Cercate di capire che cosa è successo.

8. Cambiate le linee 100 e 110 nel seguente modo

```
100 LET A=3*3*3
110 LET B=3^3
```

Eseguite il programma e riportate i risultati.

____

9. Modificate le linee 100 e 110 così:

```
100 LET A=2*2*2*2
110 LET B=2^4
```

Fate funzionare il programma. Che cosa è successo?

____

A cosa serve in Basic il simbolo "^"?

____

10. Cancellate lo schermo ed eliminate il programma dalla memoria. Inserite il seguente programma:

```
100 LET A=4+2*6/3
110 LET B=(4+2)*6/3
120 LET C=4+(2*6)/3
130 LET D=4+2*(6/3)
140 PRINT A
150 PRINT B
160 PRINT C
170 PRINT D
180 END
```

Le due cose importanti di questo programma sono (1) l'ordine in cui le operazioni vengono eseguite e (2) il significato delle parentesi. Se guardate attentamente, è chiaro che, in ognuno dei calcoli alle linee 100, 110, 120 e 130, intervengono gli stessi numeri. La sola differenza sta nel modo in cui questi numeri sono raggruppati. Eseguite il programma e segnate i risultati.

Studiate il programma ed i numeri stampati dal computer finché non vi rendete conto di come funziona il programma. In tali situazioni, il computer segue delle regole molto precise. Se non riuscite a vedere chiaramente quali sono, non preoccupatevi; tratteremo questo argomento nella discussione.

11. Cancellate lo schermo con il tasto CLR. Liberate la memoria con il comando NEW. Ora inserite il seguente programma:

```
100 LET A=3*100
110 LET B=3*100*100*100
120 LET C=3*100*100*100*100*
100
130 PRINT A
140 PRINT B
150 PRINT C
160 END
```

Nella linea 120 siete andati a capo. Quando si va a capo il computer non manda il simbolo > nel lato sinistro dello schermo. Questo indica che la linea è una continuazione della linea precedente. Eseguite il programma e annotate quanto avviene.

Potete spiegare le forme differenti in cui i numeri sono stati stampati? (Suggerimento: contate il numero degli zeri nei moltiplicatori alle linee 100, 110 e 120 del programma.)

12. Cambiate le prime tre linee del programma nel seguente modo:

```
100 LET A=3/100
110 LET B=3/(100*100*100)
120 LET C=3/(100*100*100*100
*100)
```

Nuovamente alla linea 120 siete andati a capo. Avete visto cosa è successo?
Eseguite il programma e annotate i risultati.

Cercate anche in questo caso di interpretare i risultati. Contate gli zeri nei denominatori alle linee 100, 110 e 120.

13. Se in un numero compare una E, che significato ha? Spiegatelo con parole vostre.

Se non capite ancora perfettamente il significato della notazione esponenziale, non agitatevi! Ritorneremo sull'argomento più avanti.

14. Procuratevi una cassetta non incisa e mettetela nel registratore collegato al computer. (Se non avete un registratore passate alla discussione.) Se avete qualche difficoltà nel collegare il registratore al computer consultate il manuale. Ricordate che avete un programma in memoria. Listate il programma per essere sicuri che ci sia effettivamente. Ora scrivete

```
SAVE CS1
```

Il CS1 si riferisce al registratore n. 1. Che cosa è successo?

Bene, eseguite le istruzioni che appaiono sullo schermo.

15. Se avete eseguito tutte le istruzioni in modo corretto il programma verrà registrato su nastro. Quando la registrazione è finita, il computer mostra il seguente messaggio

```
* CHECK TAPE (Y OR N)?
```

Supponiamo di voler controllare il nastro. Premete il tasto Y che sta per YES e eseguite le istruzioni.

---

16. Ora avete registrato un programma sul nastro a cassetta. Vedremo come ricaricare il programma nel computer. Per prima cosa cancellate il programma in memoria col comando NEW.
Ora scrivete

```
OLD CS1
```

Che cosa è successo?

____

Seguite le istruzioni che appaiono sullo schermo finché il programma è caricato. Per essere sicuri che tutto è andato per il verso giusto, listate il programma. Quando il programma è stato nuovamente introdotto dal nastro magnetico al computer potete usarlo come se fosse stato scritto con la tastiera.

17. Questo completa per ora il lavoro al computer. Togliete la cassetta, scrivete BYE, spegnete il computer e passate al prossimo paragrafo.

DISCUSSIONE

Nella parte pratica sono stati discussi molti punti importanti. Probabilmente non avete trovato grosse difficoltà in quegli esercizi, ma questo non vi deve far ignorare i concetti fondamentali che stanno sotto. La mancanza di comprensione a questo punto tornerà a perseguitarvi più avanti nel libro. Tratteremo quindi ognuno degli argomenti del capitolo per assicurarci che siano compresi a fondo.

Aritmetica del computer

Abbiamo a che fare con cinque operazioni aritmetiche. Queste sono addizione, sottrazione, moltiplicazione, divisione ed elevamento a potenza. Le prime quattro vi sono certamente familiari, per quanto riguarda l'ultima (esponenziazione) non lasciatevi spaventare dal nome altisonante che porta. Diamo un'occhiata ad ognuna di queste operazioni e vediamo come vengono trattate dal computer. L'addizione e la sottrazione sono fatte esattamente come vi aspettereste. I simboli usati per definire le operazioni ("+" e "–") hanno lo stesso significato per il computer che in matematica.
La moltiplicazione è trattata sul calcolatore allo stesso modo che in aritmetica, ma ha un simbolo differente che la definisce, il carattere "*". Così 2*3 è 6. A*B indica al computer di prendere in considerazione i numeri memorizzati in A e B e poi moltiplicarli. Generalmente per la moltiplicazione si usa il simbolo "×". Tuttavia poiché la × può essere un nome di variabile in Basic non possiamo usarla per la moltiplicazione. È per questa ragione che si usa il simbolo "*".

La divisione è indicata col simbolo "/". A/B significa dividere il numero che si trova nella locazione A per quello contenuto in B. Similmente 8/2 significa dividere 8 per 2.
Infine, l'operazione esponenziale è definita dal simbolo "^". Esponenziazione vuol dire "elevare a potenza". Perciò 3^4 significa "3 elevato alla quarta", che a sua volta vuol dire 3 moltiplicato per se stesso quattro volte, dando come risultato 81.
Dobbiamo stare molto attenti a capire l'ordine in cui il computer esegue le operazioni aritmetiche. Considerate la seguente espressione:

2+3^2/5-1

Se il computer analizzasse l'espressione da sinistra, eseguendo le operazioni man mano che le incontra, il risultato sarebbe 2 più 3 (cioè 5), elevato al quadrato (25), diviso per 5 (quindi 5), meno 1, per ottenere 4. Tuttavia, supponiamo che la somma e la sottrazione siano effettuate per prime, poi l'esponenziazione ed infine la moltiplicazione e la divisione. Questo darebbe 5 elevato al quadrato (25), diviso per 4, cioè 6.25.
Chiaramente, potremmo proseguire con altre regole per l'ordine delle operazioni aritmetiche, e potremmo ottenere risposte differenti ogni volta. Il punto è che, in Basic, ci sono delle regole ben precise per l'ordine e la priorità delle operazioni aritmetiche, e le dobbiamo capire. Esse sono le seguenti.
L'ordine delle operazioni è da sinistra a destra usando le regole di priorità che seguono.
La priorità per le operazioni aritmetiche è: (1) esponenziazione, (2) moltiplicazione e divisione, (3) addizione e sottrazione.

Ora, tornando al nostro esempio:

2+3^2/5-1

esaminiamo l'espressione da sinistra a destra per cercare esponenziazioni. Poiché ce n'è una, indicata da (3^2), viene eseguita per prima. Ora l'espressione è

2+9/5-1

Esaminiamola ancora da sinistra a destra, cerchiamo altre esponenziazioni e, dato che non ce n'è nessuna, cerchiamo operazioni con la prossima più alta priorità (moltiplicazione e divisione). Quindi adesso viene eseguita la divisione, con il seguente risultato:

2+1.8−1

Poiché nella espressione non ci sono altre moltiplicazioni o divisioni, la rianalizziamo in cerca di addizioni e sottrazioni. L'addizione dà:

3.8−1

e la sottrazione finale fornisce la risposta 2.8.
Rivedete le regole per l'ordine e la priorità finché non diventano naturali. Riparleremo di queste regole dopo aver discusso l'uso delle parentesi nel prossimo paragrafo.
Nei confronti delle operazioni aritmetiche va segnalato un punto molto importante valido per ogni computer.
In matematica esistono numeri decimali illimitati. Ad es. 1/3 corrisponde a 0.333333... e così via, all'infinito.
Il computer può considerare solo un numero limitato di cifre dopo la virgola. Il TI home computer, ad esempio, scrive 1/3 come 0.3333333333. In questo numero compaiono un bel po' di 3 dopo la virgola ma non certo una quantità infinita! Questo sta a significare che il computer nell'affrontare i calcoli aritmetici usa valori approssimativi e oltre a ciò c'è anche il fatto che l'aritmetica del computer è in base 2 invece che in base 10. Questo porta ad una conversione degli errori. È una conseguenza di quanto abbiamo detto il fatto che i risultati del computer sono molto precisi ma a volte non esattamente uguali a quanto ci aspettiamo.

Le parentesi nei calcoli

Le regole per l'ordine e la priorità dell'aritmetica non sono però l'unico problema. Talvolta bisogna tenere conto di altri fattori. Per capire questo consideriamo un esempio più complicato:

```
┌───────── B─┐
│┌──── A┐   │ ┌─C─┐
((2*3+4^2)*2+5)*(3^2-4)
```

Ovviamente la differenza tra questa espressione e quella che abbiamo studiato sta nell'uso delle parentesi per raggruppare parti dell'espressione. Tratteremo dettagliatamente questo esempio per mostrare come il computer affronta un'espressione aritmetica complessa.
Il computer fa l'analisi da sinistra a destra, incontra la parentesi sinistra di B, quindi guarda all'interno per vedere se ci sono altre parentesi sini-

stre e trova quella di A. La prossima parentesi incontrata è la parentesi destra di A. A questo punto, il computer ha isolato il primo gruppo di operazioni da eseguire. Questo è

```
2*3+4^2
```

e viene valutato usando le regole di ordine e priorità. Il risultato è 22 (controllatelo). Ora il nostro problema è diventato

```
┌──B───┐┌──C──┐
(22*2+5)*(3^2-4)
```

La prossima volta che l'espressione viene analizzata, il computer isola la parentesi B, fa le operazioni contenute e riduce quindi il problema a

```
   ┌C────┐
49*(3^2-4)
```

Poiché è rimasta solo la parentesi C, il computer la risolve, ottenendo

```
49*5
```

che, dopo la moltiplicazione, dà come risposta finale 245. Così, se le parentesi sono nidificate, il computer le risolve partendo dalle più interne e lavorando da sinistra a destra. Quando viene rimossa una coppia di parentesi, le operazioni contenute sono eseguite in accordo con le regole di ordine e priorità già date.
Un'ottima regola basata sull'esperienza, che il principiante può seguire, è di usare delle parentesi in più quando non è chiaro come il computer valuterà un'espressione. Troppe parentesi non possono creare guai, ma troppo poche lo possono sicuramente.
Un'ultima cosa riguardo le parentesi è che devono essere bilanciate. Cioè ci devono essere tante parentesi aperte quante chiuse. In espressioni complicate dovreste sempre contare il numero di parentesi aperte e chiuse per essere sicuri che è uguale.
Questo non garantisce la correttezza della loro disposizione, ma vi farà trovare degli errori ovvi dati dalla mancanza di parentesi.

Numeri in notazione esponenziale

I numeri vengono stampati dal Basic in forme differenti. In particolare i numeri sono talvolta scritti in quella che è conosciuta come "notazione esponenziale".
Esempi di questa notazione sono 2.456E+06 o 6.032E−14. Torneremo ai concetti introdotti nella parte pratica per chiarire l'idea di notazione esponenziale.
È facile vedere perché questa notazione particolare è richiesta sia per numeri molto grandi che molto piccoli. Un numero viene stampato dal computer con dieci cifre, ad es. 1.853695325. Sorge un problema se vogliamo che il computer stampi un numero come 4681063270000000, che richiederebbe 16 cifre. Il computer lo stamperà così: 4.68106E+15, indicante che il punto decimale deve essere spostato di quindici posizioni alla destra della sua posizione attuale. Notate come E+15 occupi le ultime quattro posizioni delle dieci usate normalmente per mostrare un numero. Un numero come 8956000000000 verrà stampato 8.956E+13. E+13 indica che il punto decimale va spostato di tredici posti verso destra.
In ogni caso non vengono mai usati più di dieci caratteri per un numero (incluse le quattro posizioni per la notazione esponenziale). Allo stesso modo possiamo anche esprimere numeri molto piccoli. Ad esempio il computer scriverà il numero 0.0000000006835984 nella forma 6.835984E−10. E−10 significa che il punto decimale va spostato di dieci posti a sinistra. La tavola che segue vi aiuterà a capire come convertire i numeri dalla notazione decimale a quella esponenziale e viceversa.

Forma decimale	Notazione esponenziale
2630000	2.63E+06
263000	2.63E+05
26300	2.63E+04
2630	2.63E+03
263	2.63E+02
26.3	2.63E+01
2.63	2.63
0.263	2.63E−01
0.0263	2.63E−02
0.00263	2.63E−03
0.000263	2.63E−04
0.0000263	2.63E−05
0.00000263	2.63E−06

Per passare da notazione decimale a esponenziale, contate il numero di posti di cui va spostato il numero decimale affinché ci sia una sola cifra

alla sua sinistra. Questo numero è quello che segue la E nella notazione esponenziale. Se dovete muovere il punto decimale verso sinistra il segno che segue la E sarà "+", se dovete muovere il punto decimale verso destra il segno che segue la E sarà "–". Per cambiare dalla notazione esponenziale alla decimale, guardate il segno che segue la E. Se il segno è "+", muovete il punto decimale verso destra di tanti posti quanti indicati dal numero che segue la E. Se il segno è "–", spostate il punto decimale verso sinistra. La notazione esponenziale non è qualcosa di cui preoccuparsi poiché la userete raramente quando scriverete dei programmi. Il motivo principale per cui è stata spiegata è che il computer potrebbe stampare dei numeri in questa notazione ed è utile che voi sappiate interpretarla.

Uso del registratore

Se ogni volta che accendiamo il computer dovessimo scrivere i programmi che vogliamo usare ci resterebbe poco tempo per lavorare. Una delle caratteristiche utili del TI home computer è l'attacco per un registratore a cassette che può essere usato per memorizzare i programmi. Una volta che un lungo programma è stato scritto e corretto, non vogliamo dover ripetere il procedimento ogni volta che ci serve. I programmi possono essere memorizzati su cassette e successivamente letti e riutilizzati ogni volta che si vuole.
Prima di discutere i comandi di sistema per la scrittura e lettura dei programmi con il registratore, ci fermeremo a considerare alcuni fatti quasi ovvi riguardanti le cassette. Primo, se registriamo un programma sopra un altro qualsiasi registrato precedentemente l'informazione precedente sarà persa. Quindi, se una cassetta contiene già dei programmi dovremo stare attenti a posizionare il nastro in modo che le informazioni da registrare vengano poste su una parte di nastro libera. Un'altra importante considerazione è che molte cassette di breve durata con un solo programma per nastro sono meglio di un lungo nastro con molti programmi. In un lungo nastro è difficile trovare la posizione di un particolare programma a meno che non si possieda un registratore costoso munito di contatore digitale. Il modo più semplice per risolvere il problema consiste nell'usare nastri brevi e registrare un singolo programma per ogni nastro. Un'ultima raccomandazione: usate nastri di buona qualità poiché nastri di cattiva qualità possono aumentare la probabilità di errori nella registrazione.
Vediamo ora come scrivere un programma sul registratore. Ovviamente tale registratore dovrà essere connesso in modo corretto al computer. Inoltre nella memoria deve essere presente un programma che vogliamo registrare. Si inizia scrivendo

```
SAVE CS1
```

CS1 si riferisce al registratore n. 1. Identificando il mezzo di output, noi possiamo collegare al computer più unità nello stesso tempo. In ogni caso, non appena avrete dato il precedente comando, il computer mostrerà il seguente messaggio

```
* REWIND CASSETTE TAPE CS1
  THEN PRESS ENTER
```

Questa istruzione viene data perché siate sicuri che il nastro si trovi nella giusta posizione. Se nel nastro non c'è alcun programma riavvolgetelo. Se ci sono già dei programmi registrati mettete il nastro all'inizio della porzione libera. In entrambi i casi, quando avete posizionato il nastro, premete il tasto ENTER. A questo punto il computer mostrerà il messaggio

```
* PRESS CASSETTE RECORD CS1
  THEN PRESS ENTER
```

Dopo aver premuto il tasto REC del registratore ed ENTER, il computer inizia la registrazione del programma in memoria sul nastro. Quando ciò avviene è visibile il messaggio

```
* RECORDING
```

Quando la registrazione è finita vedrete la scritta

```
* PRESS CASSETTE STOP CS1
  THEN PRESS ENTER
```

Seguite le istruzioni e premete lo STOP del registratore. In tutte queste istruzioni lo scopo del tasto ENTER è di permettere al computer di riconoscere che avete eseguito quanto richiesto. Dopo la registrazione il computer chiede

```
* CHECK TAPE (Y OR N)?
```

Se premete il tasto N (no), tornate nel Basic. Se premete Y (sì) il computer vi darà le istruzioni per leggere il programma dal nastro e confrontarlo con il programma in memoria. È una buona abitudine eseguire sempre questo controllo. I messaggi che accompagnano questa operazione sono

```
* REWIND CASSETTE UNIT CS1
  THEN PRESS ENTER
```

e

```
* PRESS CASSETTE PLAY CS1
  THEN PRESS ENTER

* CHECKING
```

Supponendo che non venga rilevato alcun errore, il computer mostrerà la seguente scritta

```
* DATA OK

* PRESS CASSETTE STOP CS1
  THEN PRESS ENTER
```

Se vengono trovati errori, vedrete uno dei seguenti messaggi

```
* ERROR - NO DATA FOUND

* ERROR DETECTED IN DATA
```

Subito dopo seguiranno questi altri

```
PRESS R TO RECORD CS1
PRESS C TO CHECK
PRESS E TO EXIT
```

Se premete R, l'intero processo di registrazione viene ripetuto dall'inizio. C causa il ripetersi del controllo. Infine, se premete E, tornate al Basic. Una analoga procedura viene seguita per caricare nel computer un programma letto da nastro. Ponete la cassetta nel registratore, cancellate la memoria del computer e scrivete

```
OLD CS1
```

Il computer risponderà in questo modo

```
* REWIND CASSETTE TAPE CS1
  THEN PRESS ENTER
```

Dopo aver eseguito l'istruzione vi apparirà il messaggio

```
* PRESS CASSETTE PLAY CS1
  THEN PRESS ENTER
```

E dopo ancora la scritta

```
* READING
```

che indica che il programma viene letto dal nastro.
Dopo il caricamento del programma, se non sono stati trovati errori, vedrete

```
* NO ERROR DETECTED

* PRESS CASSETTE STOP CS1
  THEN PRESS ENTER
```

A questo punto il programma è pronto per l'uso. Se invece sono stati riscontrati degli errori durante la lettura dal nastro, apparirà uno dei seguenti messaggi

```
* ERROR - NO DATA FOUND

* ERROR DETECTED IN DATA
```

Vi vengono date le seguenti alternative:

```
PRESS R TO READ
PRESS E TO EXIT
```

Se premete R l'intero processo di lettura viene ripetuto dall'inizio. Se premete E, tornate al Basic.
Potrebbe sembrare che per registrare un programma su nastro e per caricarlo poi nel computer sia necessario seguire molte istruzioni dettagliate. Tuttavia, dopo che avete scritto SAVE CS1 per iniziare il processo di registrazione, o OLD CS1 per iniziare quello di lettura, tutte le istruzioni necessarie vengono mostrate sullo schermo. Dopo aver eseguito tali operazioni alcune volte non incontrerete alcun problema.
Un ultimo commento finale riguarda le caratteristiche del registratore. Generalmente si incontrano maggiori problemi nella fase di lettura dei programmi piuttosto che in quella di registrazione. In questi casi, se vengono rilevati errori nella fase di caricamento di un programma, riprovate più e più volte finché l'operazione non avrà successo.

ESERCIZI

Gli esercizi che seguono vi serviranno per controllare se avete imparato bene i punti chiave e gli argomenti del capitolo. Controllate le vostre risposte con quelle date alla fine del libro.

1. Scrivete i simboli usati per indicare le seguenti operazioni aritmetiche nelle espressioni Basic: sottrazione, moltiplicazione, addizione, esponenziazione e divisione.

---

2. Quando, nel valutare delle espressioni aritmetiche, c'è una priorità di operazioni, qual è questa priorità?

---

3. Nell'analisi delle espressioni aritmetiche, il computer fa la ricerca in una direzione specifica. Qual è questa direzione?

---

4. Scrivere un'istruzione Basic per valutare la seguente espressione. Date come numero di linea 100.

$$A = (4 + 3B/D)^2$$

---

5. Se il seguente programma viene eseguito che cosa verrà stampato?

```
100 LET A=2
110 LET B=3
120 LET C=(A*B+2)/2
130 PRINT C
140 END
```

---

6. Convertite i seguenti numeri in notazione esponenziale: (a) 567300000000000 e (b) 0.000003814275168.

---

7. Convertite i seguenti numeri in notazione decimale: (a) 7.258E+06 e (b) 1.437E–03.

---

8. Nell'espressione sottostante dite qual è l'ordine in cui vengono eseguite le operazioni dal computer.

```
100 LET A=(6/3+4)^2
```

9. Come si registra un programma su una cassetta?

____

10. Come si legge un programma da una cassetta?

____

CAPITOLO CINQUE

Input, output e semplici applicazioni

ARGOMENTI

In questo capitolo tratteremo il problema della scrittura di programmi che svolgono certi compiti. Allargheremo anche la nostra conoscenza del Basic spiegando alcuni dettagli relativi all'input e all'output (inserimento e uscita dei dati). Gli argomenti sono i seguenti.

Come inserire dei numeri in un programma

Ci sono solo tre modi mediante i quali è possibile inserire dei numeri nel computer. Dobbiamo imparare quali sono.

Stampa di variabili e stringhe

Quando i dati richiesti sono stati colcolati, devono essere stampati. È possibile ottenere l'output in vari modi. Solitamente si vuole un output misto di stringhe e numeri. L'output delle stringhe è trattato essenzialmente allo stesso modo dei numeri ma richiede particolare attenzione.

La spaziatura dell'output

Nell'argomento precedente eravamo interessati all'output di numeri e stringhe di caratteri. Qui abbiamo a che fare con la spaziatura di quell'output.

L'istruzione REM

Il programmatore intelligente include nei programmi dei commenti che aiutano a spiegare o interpretare quello che viene fatto. L'istruzione REM dà questa possibilità.

Semplici applicazioni

Il nostro scopo finale è di imparare a scrivere e correggere dei programmi. In questo capitolo inizieremo con alcuni semplici esercizi di programmazione.

PRATICA SUL CALCOLATORE

Passiamo alla parte pratica.

1. Accendete il vostro calcolatore ed inserite il seguente programma:

```
100 INPUT A
110 INPUT B
120 INPUT C
130 LET D=A+B+C
140 PRINT D
150 END
```

   Che cosa pensate succederà se lo eseguiamo?

   ____

   Fate funzionare il programma. Quando appare il primo punto interrogativo (segnale di input per A), inserite 2. Similmente, quando compare il secondo punto interrogativo, date 3, ed infine, all'ultimo segnale di input, inserite 5. Riportate quello che è successo.

   ____

2. Notate che nel programma del passo 1 ci sono tre istruzioni INPUT (linee 100, 110 e 120). Scrivete

```
100
110
```

Che cosa farà questo programma?

___

Scrivete

```
120 INPUT A,B,C
```

Listate il programma. Che cosa è successo?

___

3. Eseguite il programma, e quando compare il segnale di input (?), inserite

```
2,3,5
```

Che cosa è accaduto?

___

È possibile in un programma avere un input unico di più variabili?

___

4. Eseguite ancora il programma, e questa volta, quando compare il segnale di input, scrivete

```
2,3
```

Che cosa è successo?

___

Qual è il problema?

___

5. Il computer sta ancora aspettando l'entrata dei dati. Questa volta scrivete

```
2,3,5,1
```

Che cosa è accaduto?

---

6. Si possono inserire più numeri di quelli richiesti da un'istruzione INPUT?

Che cosa succede se lo fate?

---

7. Potete inserire meno numeri di quelli richiesti in un'istruzione INPUT?

Che cosa succede se lo fate?

---

8. Scrivete

```
120 READ A,B,C
```

Listate il programma. Che cosa è successo?

Eseguite il programma e annotate i risultati.

---

9. Ora scrivete

```
125 DATA 2,3,5
```

e listate il programma. Che cosa è accaduto?

---

10. Fate funzionare il programma e annotate quello che è successo.

Basandovi su quello che avete appena visto, ogni volta che un programma contiene un'istruzione READ, ci deve essere un altro tipo di istruzione nel programma. Qual è questa istruzione?

---

11. Indicate due metodi differenti (che non siano l'istruzione di assegnazione) per inserire dei numeri in un programma. (Suggerimento: vedere i passi 2 e 8.)

---

12. Listate il programma in memoria. Cancellate l'istruzione DATA e poi scrivete

```
145 DATA 2,3,5
```

Poiché non possiamo correggere i numeri di linea dobbiamo riscrivere la linea con un nuovo numero. Fate ancora la lista del programma. Che cosa è successo?

---

13. Eseguite il programma e riportate l'output.

Vi sembra che la posizione dell'istruzione DATA nel programma sia importante?

---

14. Liberate la memoria con il comando NEW ed inserite il programma che segue.

```
100 READ A,B
110 LET C=A/B
120 PRINT C
130 GOTO 100
140 DATA 2,1,6,2,90,9,35,7
150 END
```

Che cosa pensate succederà se lo eseguite?

Provate e guardate se avevate detto giusto. Riportate l'output.

Il messaggio DATA ERROR è associato all'istruzione READ o all'istruzione DATA?

---

15. Togliete dal programma l'istruzione DATA alla linea 140. Ora inserite

```
105 DATA 10,2
115 DATA 100,50
125 DATA 50,5
```

Listate il programma. Che cosa è successo?

---

16. Se eseguiamo il programma, che cosa pensate che verrà stampato?

Fate funzionare il programma e guardate se avevate detto giusto.

---

17. Si possono avere in un programma più istruzioni DATA?

Vi sembra che la posizione delle istruzioni DATA abbia qualche significato?

---

18. Cancellate il programma dalla memoria. Inserite il seguente programma:

```
100 LET A=10
110 PRINT A
120 END
```

Che cosa accadrà se lo eseguite?

Fate funzionare il programma e segnate quello che è successo.

---

19. Ora scrivete

```
110 PRINT "A"
```

e listate il programma. Che cosa è successo?

Che cosa succederà se lo eseguiamo?

Eseguite il programma e segnate quello che il computer ha stampato.

---

20. Scrivete

```
110 PRINT "CANE DA CACCIA = ";A
```

e listate il programma. Che cosa pensate che accadrà se ora facciamo funzionare il programma?

Eseguite il programma e annotate i risultati.

---

21. Ora tentiamo qualcosa di diverso. Scrivete

```
105 LET B=2
110 PRINT "B = ";A
```

Listate il programma e studiatelo attentamente. Se lo eseguiamo, che cosa pensate che succederà?

Provate e guardate se avevate detto giusto. Riportate l'output.

---

22. Scrivete

```
95 REM  PROGRAMMA DIMOSTRATI
VO
```

Listate il programma. Che cosa è successo?

Fate funzionare il programma. Qual è stato l'output?

L'istruzione REM alla linea 95 ha qualche effetto sul programma?

---

23. Cancellate il programma dalla memoria ed inserite il programma che segue:

```
100 REM  PROGRAMMA DI CON-
110 REM  VERSIONE DA LIBBRE

120 REM   IN GRAMMI
130 PRINT "NUMERO DI LIBBRE"
:
140 INPUT P
150 LET G=454*P
160 PRINT P;" LIBBRE SONO"
170 PRINT G;" GRAMMI"
180 GOTO 130
190 END
```

Listate il programma e controllatelo per vedere se è corretto. Studiatelo attentamente e tentate di indovinare che cosa succederà se lo eseguiamo. Ora eseguitelo. Quando compare il segnale di input, inserite un numero qualsiasi. Osservate quello che viene stampato. Ripetete questo processo parecchie volte, poi fate uscire il computer dal ciclo di input. Ricordate che questo si ottiene premendo il tasto SHIFT-C (FCTN-4). Che scopo ha l'istruzione REM?

---

24. Proviamo ad usare l'input in modo diverso.

```
120
130 INPUT "NUMERO DI LIBBRE"
:P
```

Listate il programma e osservate attentamente i cambiamenti. Notate la stringa di caratteri nell'istruzione INPUT alla linea 130. Che cosa pensate che accadrà?

Eseguite il programma e annotate cosa è successo.

Questa possibilità di avere un segnale in corrispondenza di un'istruzione **INPUT** è un'utile caratteristica del TI Basic.

25. Scrivete

```
115 INPUT P
120 PRINT "NUMERO DI LIBBRE"
;
130
170 GOTO 115
```

Listate il programma. Che cosa è successo?

____

Funzionerà il programma in questa forma?

____

Fatelo funzionare, e al segnale di input scrivete **1**. Che cosa è successo?

____

Fate uscire il programma dal ciclo di input.

26. Sperimentate questo programma un po' di più. Cancellate il programma dalla memoria e scrivetelo nuovamente modificato come segue:

```
100 REM  PROGRAMMA DI CON-
110 REM  VERSIONE DA LIBBRE
120 REM  IN GRAMMI
130 PRINT "NUMERO DI LIBBRE"
;
140 INPUT P
150 PRINT P;" LIBBRE SONO"
160 PRINT G;" GRAMMI"
170 LET G=454*G
180 GOTO 120
190 END
```

Il programma può essere eseguito in questa forma?

____

Eseguite il programma e, all'input, scrivete 2. Che cosa è successo?

____

Spiegate con parole vostre cosa vi è di sbagliato. Ricordatevi che se

una variabile non viene inizialmente definita nel vostro programma il computer la porrà uguale a 0.

---

27. Fate uscire il computer dal ciclo di input. Cancellate il programma in memoria ed inserite il seguente

```
100 READ A
110 PRINT A
120 GOTO 100
130 DATA 10,12,9,73,60,82
140 END
```

Fate girare il programma e segnate che cosa è successo. Ponete una particolare attenzione alla spaziatura dei numeri.

---

28. Aggiungete una virgola dopo la A nella linea 110. Eseguite il programma e annotate quello che è successo.

---

29. Ora sostituite la virgola dopo la A con un punto e virgola. Fate funzionare il programma e annotate quello che è successo.

---

30. Se una variabile, in un'istruzione PRINT, non è seguita da nessun segno di interpunzione, che cosa succede dopo che il numero è stato stampato? (Suggerimento: vedi passo 27.)

E supponendo che la variabile sia seguita da una virgola?

Cosa succederà se la variabile è seguita da un punto e virgola?

---

31. Cancellate il programma dalla memoria. Inserite il seguente programma:

```
100 LET A=10
110 READ B
```

```
120 PRINT TAB(A);B;
130 LET A=A+10
140 GOTO 110
150 DATA 1,2,3
160 END
```

Eseguite il programma e segnate cosa è accaduto.

---

32. Sostituite A+10 nella linea 130 con A+5. Fate funzionare il programma e annotate quello che è successo. Fate anche qui particolare attenzione alla spaziatura.

---

33. Ora cambiate, nella linea 130, A+5 con A+3. Eseguite il programma e segnate quello che è successo.

---

34. Cosa controlla il TAB nell'istruzione di stampa?

---

35. Questo conclude per ora la parte pratica. Scrivete BYE, spegnete il vostro computer e passate alla discussione.

DISCUSSIONE

In questo capitolo abbiamo cominciato ad allontanarci dal semplice controllo del computer. Invece, ci concentreremo di più sulla scrittura e correzione dei programmi. A molti studenti questo non viene naturale, quindi l'argomento sarà trattato con molta attenzione, sia ora che nei prossimi capitoli.

Come inserire dei numeri in un programma

Nel capitolo 3 abbiamo visto un modo per inserire dei numeri in un programma. Si faceva assegnando dei valori a una variabile nel programma stesso. Per esempio,

```
100 LET A=6
```

introduce il valore 6 nel programma e memorizza il numero sotto il nome di variabile A. Questo metodo ha delle limitazioni. Dobbiamo esaminare altri modi con cui introdurre dei numeri nei programmi.
Cominciamo col considerare l'istruzione INPUT e come si usa. Un esempio potrebbe essere

```
260 INPUT G
```

Quando il calcolatore esegue questa linea, stamperà un punto interrogativo come segnale che sta aspettando un input dalla tastiera; poi si fermerà e aspetterà che voi inseriate il numero. Nel caso sopra, il numero dato sarà assegnato alla variabile G.
Con una singola istruzione INPUT si può richiedere più di una variabile; ad esempio

```
420 INPUT A,B,C,D
```

In questo caso viene stampato lo stesso segnale di input (il punto interrogativo), ma ora il calcolatore sta aspettando l'inserimento di quattro numeri separati da virgole. Se vengono inseriti soltanto tre numeri e poi si preme ENTER, il computer manderà un messaggio d'errore poiché non sono stati dati i valori attesi e vi chiederà di provare un'altra volta. Se inizialmente vengono inseriti più di quattro numeri, il computer scriverà un messaggio d'errore come sopra e aspetterà che riscriviate i valori di input.
È opportuno generalmente far precedere un'istruzione INPUT da un messaggio che spieghi che tipo di variabile va introdotta. Questa scritta può essere inclusa nello stesso comando di INPUT. Un esempio di ciò potrebbe essere

```
150 INPUT "SCRIVI IL PESO":W
```

Se viene eseguita questa istruzione, apparirà il messaggio SCRIVI IL PESO. Poi il computer si fermerà aspettando che voi scriviate il valore di W. Usando l'istruzione INPUT in questa forma non apparirà alcun punto di domanda. Notate i due punti che separano la stringa di caratteri dalla variabile di input. Questi due punti devono essere presenti altrimenti compare un messaggio di errore. Un ultimo commento sull'istruzione INPUT. Potete chiedere in input sia variabili numeriche che di stringa. Un esempio potrebbe essere

```
130 INPUT A,B$
```

In questo caso il computer aspetterà che voi scriviate un numero, una

virgola, e una stringa di caratteri. È importante che ci sia una corrispondenza tra i dati che vengono introdotti e quelli che il computer si aspetta. Se nell'esempio precedente aveste scritto

```
2L3,CASA
```

il computer avrebbe rilevato un errore e vi avrebbe chiesto di inserire nuovamente i dati. E ciò a causa della L nel numero. Come abbiamo già puntualizzato, un errore comune è quello di battere "L" invece di "1". In questo caso, si era supposto di scrivere il numero 213, ma il computer ha trovato la L che non può essere in un numero. Fate quindi attenzione a scrivere numeri quando sono richiesti dei numeri e una stringa di caratteri quando è richiesta una stringa, in questo modo non avrete problemi. Se commetterete un errore il computer ve lo dirà!
L'ultimo metodo per dare dei numeri al calcolatore è mediante le istruzioni READ e DATA. L'istruzione

```
100 READ A,B,C,D
```

è trattata dal calcolatore allo stesso modo di un'istruzione INPUT, con due eccezioni. Primo, il computer non si ferma. Non ce n'è bisogno, come si vedrà. La seconda eccezione è che i numeri richiesti vengono letti dall'istruzione DATA contenuta nel programma e non inseriti dall'utente con la tastiera in risposta ad un segnale di input. Per illustrare le istruzioni READ e DATA, considerate il seguente programma:

```
100 READ A,B,C,D
110 LET E=A+B+C+D
120 PRINT E
130 DATA 25,3,17,12
140 END
```

Il programma legge quattro numeri dall'istruzione DATA e stampa la loro somma. La posizione dell'istruzione DATA nel programma non conta, però l'istruzione finale deve ancora essere END. Ci possono essere più istruzioni DATA e non occorre che siano raggruppate nello stesso punto del programma. Man mano che i numeri sono richiesti dalle istruzioni READ, essi vengono presi in ordine dalle istruzioni DATA, iniziando da quella con il numero più basso. Se, dopo aver usato tutti i numeri delle istruzioni DATA, ne vengono richiesti altri, il computer scriverà il messaggio DATA ERROR e si fermerà. D'altro canto è possibile che un programma non usi tutti i numeri delle istruzioni DATA, nel qual caso non comparirà alcun messaggio d'errore.
Riassumendo ci sono tre metodi con i quali si possono introdurre numeri nei programmi. Essi sono (1) l'istruzione di assegnazione, (2) l'istruzione INPUT e (3) le istruzioni READ e DATA. A seconda dei casi ognuno

di questi metodi presenta dei vantaggi. Capirete meglio i vantaggi e gli svantaggi di ogni metodo man mano che imparerete a programmare.

Stampa di variabili e stringhe

L'output dal computer è molto semplice. Il computer può stampare sia il valore numerico di una variabile (un numero) che una stringa di caratteri. Come esempio supponiamo di avere una variabile chiamata X e che il numero 2 sia contenuto in quella locazione. Il programma

```
100 LET X=2
110 PRINT "X"
120 PRINT X
130 END
```

mostra la differenza tra i due tipi di output. La linea 110 stampa il carattere X, poiché questo è racchiuso tra virgolette. La linea 120 stampa 2 poiché questo è il numero contenuto nella locazione X.
La regola è chiara. Una qualsiasi successione di caratteri racchiusa tra virgolette è chiamata stringa. Le stringhe vengono stampate esattamente come sono. Il calcolatore non cerca di analizzarle o di scoprire che cosa contengono. Se una variabile, in un'istruzione PRINT, non è racchiusa tra virgolette, il computer stampa il valore numerico di quella variabile.
In un'istruzione PRINT è possibile fare dei calcoli. Così

```
100 PRINT A+B+C,D
```

farà stampare al computer la somma dei numeri memorizzati in A, B e C, seguiti dal numero contenuto in D.

La spaziatura negli output

La versione del Basic fornita nel TI home computer contiene un meccanismo interno di spaziatura standard, il quale, su una linea, stampa due numeri ugualmente distanziati. Questa spaziatura standard viene usata dal computer quando delle quantità in un'istruzione PRINT sono distanziate da virgole. La virgola indica al computer di spostarsi alla prossima posizione di stampa sulla linea. Se il computer è già alla seconda posizione della linea ed incontra una virgola in un'istruzione PRINT, va a capo e stampa il numero sulla prima posizione della prossima linea. Quindi

```
100 PRINT A,B,C
```

causerebbe la stampa dei valori numerici di A e B nelle due posizioni standard. Il valore C sarebbe stampato nella linea successiva al di sotto del valore di A.
Un altro tipo di spaziatura è prodotto dal punto e virgola tra le variabili, ad esempio

```
100 PRINT A;B;C
```

Il punto e virgola causa una spaziatura più ristretta di quella standard ottenuta con la virgola. Tuttavia, in questo caso, la spaziatura non è sempre uniforme, poiché i numeri possono essere stampati con notazioni differenti. Ad esempio, l'istruzione

```
100 PRINT A;B
```

causa una spaziatura dell'output più ristretta di

```
100 PRINT A,B
```

Infine, possiamo controllare meglio la spaziatura su una linea usando la funzione TAB nelle istruzioni PRINT. La funzione TAB lavora nello stesso modo del tabulatore di una macchina da scrivere. Su una singola linea dello schermo del computer ci sono ventotto posizioni di stampa. L'istruzione

```
100 PRINT TAB(5);A;TAB(20);B
```

indica al calcolatore di passare alla quinta posizione di stampa, stampare il valore numerico di A, andare alla ventesima posizione di stampa ed infine stampare il valore numerico di B. È anche possibile avere un posizionamento variabile, cioè controllato dal calcolatore:

```
100 PRINT TAB(X);A
```

Qui il computer prima considera il valore di X, poi si sposta alla posizione di stampa determinata dal più grande intero minore o uguale di X (per esempio, se X = 23.14350826, il calcolatore si sposterà alla ventitreesima posizione di stampa), e poi stampa il valore numerico di A. Poiché ci sono solo ventotto posizioni di stampa in una linea, vi chiederete cosa potrebbe accadere se il computer tentasse di eseguire:

```
100 PRINT TAB(40);B
```

Accadrà che il computer comincerà a sottrarre 28 dal numero che appa-

re nella funzione TAB finché questo è minore o uguale a 28. Nel nostro esempio eseguirà la sottrazione 40−28 = 12 che è inferiore a 28. Allora il computer creerà uno spazio fino alla dodicesima posizione e poi stamperà il valore di B.
Possiamo ottenere una spaziatura verticale nell'output usando un'istruzione PRINT. Come segue

```
100 PRINT
```

Quando il computer esamina ciò che deve stampare e non trova niente, guarda se esiste la punteggiatura e non trovando niente ordina al cursore un ritorno a capo della riga successiva.
Se desideriamo due o tre linee vuote, possiamo ottenere una spaziatura verticale usando tante istruzioni PRINT vuote quante desideriamo.
Un'altra variazione dell'istruzione PRINT consiste nell'usare i due punti per separare le variabili da stampare. I due punti provocano un ritorno a capo della linea successiva. Ad esempio

```
100 PRINT A:B:C
```

e

```
100 PRINT A
110 PRINT B
120 PRINT C
```

hanno esattamente lo stesso risultato in un programma.
Con l'istruzione PRINT potete stampare stringhe di caratteri. Un esempio potrebbe essere

```
100 PRINT A$,B$
```

Se A$ e B$ sono entrambe abbastanza brevi, il computer le stamperà sulla stessa linea. Al contrario, se B$ è troppo lunga, A$ verrà stampata su una linea e B$ sulla successiva. Infine, se A$ è troppo lunga per una sola linea, il suo seguito verrà completato alla linea successiva.

L'istruzione REM

L'istruzione REM (che sta per *remark*, cioè commento) è completamente differente da quelle che abbiamo visto finora. Non appena il computer si accorge dei caratteri REM che seguono il numero di linea, ignora il resto dell'istruzione e passa alla prossima linea.

Qual è allora, lo scopo di questa istruzione, se il calcolatore la trascura? L'istruzione REM è un modo di fornire informazioni a beneficio del programmatore o di qualcuno che legga il programma. Queste informazioni rendono più facile seguire quello che sta succedendo nel programma. Il programmatore saggio userà queste istruzioni in abbondanza.
Per illustrare l'uso delle istruzioni REM vedremo due programmi. Entrambi danno lo stesso risultato, ma il secondo usa istruzioni REM per descrivere che cosa sta succedendo nel programma. Giudicate voi stessi quale programma è più facile da seguire. Senza istruzioni REM

```
100 INPUT A,B,C,D
110 LET X=(A+B+C+D)/4
120 PRINT X
130 END
```

Con istruzioni REM

```
100 REM  CALCOLO DELLA MEDIA
 DI QUATTRO NUMERI
110 REM  INPUT DEI QUATTRO N
UMERI
120 INPUT A,B,C,D
130 REM  CALCOLO DELLA MEDIA
140 LET X=(A+B+C+D)/4
150 REM  SCRITTURA DELLA MED
IA
160 PRINT X
170 END
```

Notate che nel programma le linee 100 e 110 e 150 sono più lunghe del massimo numero di ventotto caratteri che può essere contenuto in una riga. In questo caso la parte eccedente viene stampata a capo. Le linee molto lunghe vengono riportate anche in più di due linee successive sullo schermo. Se noi scriviamo programmi più complicati, vedrete che ciò accadrà con maggior frequenza. Quando le linee vengono riportate a capo, il computer non manda il segnale ">" che si trova normalmente sulla sinistra dello schermo. Un fatto che va considerato è che i numeri presenti in una linea lunga, se vengono riportati a capo, possono capitare giusto nella posizione che può farli apparire come numeri di linea di una linea successiva. In realtà ciò accade molto raramente, ma poiché potrebbe verificarsi è giusto che siate avvertiti.

ESEMPI DI PROGRAMMI

Come abbiamo detto prima, trascorreremo progressivamente più tempo sulla scrittura e correzione dei programmi. Gli esempi scelti per questo capitolo sono molto semplici, ma illustrano molto bene i concetti che

abbiamo discusso finora. Studiate ogni esempio attentamente finché non siete certi di aver capito tutti i dettagli. Potete inserire i programmi nel vostro computer ed eseguirli per verificare che funzionino come indicato.

Esempio 1 - Prezzi unitari

ıostro problema è di scrivere un programma che calcoli i prezzi unitai ɲrodotti di un supermercato. Indicheremo con T il prezzo totale del ..o, con N il numero di pezzi contenuti nel pacco e con U il prezzo unitario. Possiamo calcolare il prezzo unitario con la seguente relazione:

U = T/N

Come esempio, supponiamo che una scatola di dodici bottiglie di succo di frutta costi 6960 lire. Allora il costo unitario a bottiglia sarebbe

U = 6960/12 = 580

Vogliamo che il programma sia organizzato in modo da fornire il seguente output quando eseguito:

```
PREZZO TOTALE? 6960
QUANTI PEZZI? 12
IL PREZZO UNITARIO E' 580
```

I numeri dopo i punti interrogativi saranno inseriti in fase di esecuzione del programma. Dato un qualsiasi prezzo totale ed il numero dei pezzi, il programma dovrà calcolare e stampare il prezzo unitario corretto. Ricordate che se lo desiderate, si possono usare nomi di variabile come TOTALE, NUMERO e PREZZO UNITARIO invece di T, N e U. Esaminiamo la prima linea dell'output desiderato. C'è stampato un messaggio seguito da un punto interrogativo e da un numero dato in input con la tastiera. Questo può essere ottenuto facilmente con le seguenti istruzioni:

```
100 PRINT "PREZZO TOTALE";
110 INPUT T
```

Ricordate che il T indica il prezzo totale. Il punto e virgola alla linea 100 impedisce il ritorno del cursore sulla parte sinistra dello schermo. Le prossime due linee del programma sono scritte nello stesso stile delle prime due.

```
100 PRINT "QUANTI PEZZI ";
110 INPUT N
```

N sta per il numero di pezzi. Dobbiamo ora calcolare il prezzo unitario, che sarà chiamato U.

```
140 LET U=T/N
```

Tutto quello che rimane da fare è stampare la linea finale di output ed aggiungere l'istruzione END.

```
150 PRINT "IL PREZZO UNITARI
O E'"
160 PRINT U
170 END
```

Vediamo il programma completo.

```
100 PRINT "PREZZO TOTALE";
110 INPUT T
120 PRINT "QUANTI PEZZI";
130 INPUT N
140 U=T/N
150 PRINT "IL PREZZO UNITARI
O E'"
160 PRINT U
170 END
```

Studiate il programma per essere sicuri di aver capito lo scopo di ogni linea in relazione alla descrizione originale di quello che si voleva. Provatelo con vari dati iniziali finché non avrete compreso esattamente come lavora il programma.

Esempio 2 - Conversione di temperature

La relazione tra temperature misurate in gradi Fahrenheit e gradi Celsius è

```
C=5/9(F-32)
```

In questa espressione C sta per gradi Celsius e F per gradi Fahrenheit. Se, per esempio, F è 212, allora C varrà

```
C = 5/9(212-32) = 100
```

Come nel primo esempio, scriveremo il programma dopo aver visto come vogliamo che appaia l'output. Supponiamo che, eseguendo il programma desiderato, si voglia vedere il seguente tipico output:

```
QUANTI GRADI F
? 212
EQUIVALGONO A 100 GRADI C
```

Si noti che le prime due linee dell'output desiderato sono leggermente differenti dall'Esempio 1. In questo caso, il punto interrogativo e l'input dalla tastiera sono sulla seconda linea. Questo si ottiene omettendo il punto e virgola alla fine del primo messaggio.

```
100 PRINT "QUANTI GRADI F"
110 INPUT F
```

Ora calcoliamo il numero di gradi Celsius usando la relazione data sopra.

```
120 LET C=(5/9)*(F-32)
```

Infine stampiamo l'ultimo messaggio e la risposta.

```
130 PRINT "EQUIVALGONO A ";C
;" GRADI C"
140 END
```

La linea 130 illustra come si possono stampare stringhe di caratteri e variabili numeriche nella stessa istruzione PRINT. Poiché C non è tra virgolette, verrà stampato il valore numerico. Ecco il programma completo.

```
100 PRINT "QUANTI GRADI F"
110 INPUT F
120 LET C=(5/9)/(F-32)
130 PRINT "EQUIVALGONO A ";C
;" GRADI C"
140 END
```

Come nell'Esempio 1, potete provare questo programma usando diversi valori di F.

Esempio 3 - Pagamento mensile di un mutuo bancario

Passiamo ora ad un esempio più complicato (ed anche più utile). Vogliamo scrivere un programma che calcola le rate mensili di un mutuo bancario. La relazione da calcolare è

```
M=(PI/1200)/(1-1/(1+I/1200))^(12N)
```

In questa relazione, P è l'ammontare iniziale del capitale in lire, I è il tasso d'interesse annuale in percentuale, N è la durata del mutuo in anni ed M è la rata mensile in lire. Vogliamo che l'output, dopo l'esecuzione del programma, appaia come segue:

```
CAPITALE (L) = 50000
TASSO D'INT. (%) = 8.5
PERIODO (ANNI) = 30
RATA MENSILE (L)
384.4567450
```

Come prima, l'input dalla tastiera segue il punto interrogativo e rappresenta un caso tipico. Il pagamento mensile è mostrato come il calcolatore lo stamperà. In un prossimo capitolo, vedremo come arrotondarlo alla lira più vicina.
Ormai le prime linee del programma dovrebbero essere scritte senza difficoltà. Notate che stiamo usando i messaggi di input in modo leggermente diverso rispetto ai precedenti esempi.

```
100 INPUT "CAPITALE (L) = ":
P
110 INPUT "TASSO D'INT. (%)
= ":I
120 INPUT "PERIODO (ANNI) =
":N
```

Usando i valori di P, I e N che sono stati inseriti, dobbiamo ora calcolare il pagamento mensile. Questo sarà fatto in tre passi.

```
130 LET X=P*I/1200
140 LET Y=(1+I/1200)^(12*N)
150 LET M=X/(1-1/Y)
```

Studiate le espressioni originali e le linee 160, 170 e 180 finché non siete sicuri di aver capito come viene effettuato il calcolo. Le linee finali del programma sono

```
160 PRINT "PAGAMENTO MENSILE
 L)"
170 PRINT M
180 END
```

Ecco il programma completo.

```
100 INPUT "CAPITALE (L) = ":
P
110 INPUT "TASSO D'INT. (%)
= ":I
120 INPUT "PERIODO (ANNI) =
":N
130 LET X=P*I/1200
```

```
140 LET Y=(1+I/1200)^(12*N)
150 LET M=X/(1-1/Y)
160 PRINT "PAGAMENTO MENSILE
 (L)"
170 PRINT M
180 END
```

Questo programma vi può essere utile se state cercando una casa. Potete subito stabilire se una certa casa rientra nelle vostre possibilità economiche.

PROBLEMI

1. Scrivete un programma che legga i quattro numeri 10, 9, 1 e 2 da un'istruzione DATA, mettendoli rispettivamente in A, B, C e D; esegua la somma dei primi due e metta il risultato in S, calcoli il prodotto degli ultimi due e lo ponga in P. Poi stampi i valori di S e P sulla stessa linea.

2. Scrivete un programma che richieda l'input di quattro numeri e poi li stampi in ordine inverso. Per esempio, se voi scrivete 5, 2, 11, 12 il computer dovrà stampare 12, 11, 2 e 5. Il programma deve funzionare con qualsiasi insieme di numeri si decida di inserire. Può essere scritto in solo due linee oltre l'END.

3. Che output otterremo eseguendo il seguente programma?

```
100 READ X,Y,Z
110 DATA 2,5,3
120 LET T=X+Y*Z
130 LET S=Y^2
140 PRINT T,S
150 END
```

4. Spiegate con parole vostre che cosa fa il seguente programma.

```
100 INPUT A,B
110 LET S=A+B
120 LET T=A-B
130 LET U=A*B
140 PRINT S,T,U
150 END
```

5. Se un oggetto viene lasciato cadere vicino alla superficie della terra, la distanza che percorrerà in un tempo dato può essere determinata da

$$S = 4.9T^2$$

dove S è la distanza (in metri) e T è il tempo di caduta (in secondi). Scrivete un programma che, quando eseguito, fornisca un output simile al seguente:

```
TEMPO DI CADUTA (SEC) ? 2
LA DISTANZA PERCORSA
DALL'OGGETTO E' 64 METRI
```

6. Il volume di una scatola può essere calcolato con la formula V = LWH, dove L, W e H sono lunghezza, larghezza e altezza, rispettivamente. Ad esempio, se queste sono misurate in centimetri, il volume sarà in centimetri cubi. Vogliamo un programma che, quando eseguito, fornisca un output simile al seguente:

```
LUNGHEZZA (CM) ? 4
LARGHEZZA (CM) ? 2
ALTEZZA (CM) ? 3
IL VOLUME E' 24 CM. CUBI
```

Il programma che segue non è corretto, e non darà l'output desiderato. Che cosa c'è di sbagliato?

```
100 PRINT "LUNGHEZZA (CM)";L
110 PRINT "LARGHEZZA (CM)";W
120 PRINT "ALTEZZA (CM)";H
130 INPUT L,W,H
140 LET V=L*W*H
150 PRINT "IL VOLUME E'"
160 PRINT V
170 PRINT "CM CUBI"
180 END
```

7. Nel seguente programma, vengono inseriti due numeri, A e B, mediante un'istruzione INPUT. Il problema è di mettere le tre istruzioni mancanti in modo che, quando A e B vengono stampati, i loro valori risultino scambiati.

```
100 INPUT A,B
110
120
130
140 PRINT A,B
150 END
```

8. Supponiamo che il contachilometri della vostra macchina indichi R1 chilometri quando il serbatoio è pieno. Voi guidate finché il contachilometri segna il numero R2; a questo punto sono necessari L litri di benzina per riempire il serbatoio. Per sapere quanti chilometri avete percorso con un litro, dovete usare la relazione C = (R2-R1)/L. Scrivete un programma che fa questo calcolo per i seguenti dati:

```
R1      R2      G

21423   21493   5
05270   05504   13
65214   65559   11.5
```

9. Se una somma di denaro P è lasciata accumulare interessi ad un tasso percentuale annuo I, per N anni, il denaro aumenterà fino ad un totale T dato da

```
T = P(1+I/100)^N
```

Ad esempio, se P = 1000, I = 6%, e N = 5 anni,

```
T=1000(1+6/100)^5=1338.23
```

Scrivete un programma che, quando eseguito, dia un output simile al seguente

```
CAPITALE ? 1000
TASSO D'INT. (%) ? 6
PERIODO (ANNI) ? 5
IL VALORE TOTALE E'
1338.22558
```

10. Se una somma di denaro P è lasciata accumulare interessi ad un tasso percentuale I composto J volte all'anno, per N anni, la somma finale sarà

$$T = P(1 + I/100J)^{JN}$$

Scrivete un programma che riceva in input P, I, J ed N. Eseguitelo per calcolare il valore ottenuto investendo 1000 al tasso dell'8% per 2 anni composto: (a) annualmente, (b) semestralmente, (c) mensilmente, (d) settimanalmente ed (e) giornalmente. Se una compagnia di risparmio e prestito fa una grossa campagna pubblicitaria sul fatto che loro calcolano l'interesse ogni giorno anziché ogni settimana, vi sentireste interessati?

ESERCIZI

Gli esercizi che seguono vi permetteranno di controllare fino a che punto avete acquisito i concetti e gli argomenti di questo capitolo. Controllate le vostre risposte con quelle date alla fine del libro.

1. Quale sarà l'output, se eseguiamo il seguente programma?

```
100 LET X=1
110 PRINT X,
120 LET X=X+1
130 GOTO 110
140 END
```

____

2. Descrivete tre modi con cui si possono inserire dei numeri in un programma Basic.

____

3. In un'istruzione PRINT, come è chiamata una collezione di caratteri tra virgolette?

____

4. Che scopo ha l'istruzione REM?

____

5. Se in un programma c'è un'istruzione READ, che altro tipo di istruzione deve anche essere presente nel programma?

____

6. Che cosa succederà se viene eseguito il seguente programma?

```
100 LET X=3
110 LET Y=4
120 PRINT "Y = ";X
130 END
```

____

7. Qual è il numero standard di colonne che vengono riservate quando le quantità da stampare sono separate da virgole?

____

8. Quante istruzioni DATA ci possono essere in un programma?

____

9. A che cosa serve l'istruzione TAB?

____

10. Che cosa accadrà se il seguente programma viene eseguito?

```
100 LET A=1
110 LET B=3
120 PRINT A,B
130 PRINT A;B
140 END
```

---

11. Il programma

```
100 INPUT A,B
110 LET C=A+B
120 PRINT C
130 END
```

viene eseguito e, in risposta al segnale di INPUT, scriviamo i numeri 10, 12 e 13. Descrivete esattamente quello che succederà.

---

12. Le miglia possono essere trasformate in chilometri moltiplicando per 1.609. Quindi, 10 miglia sono 16.09 chilometri, e così via. Scrivete un programma che, quando eseguito, fornisca un output simile al seguente:

```
QUANTE MIGLIA ? 2.5
2.5 MIGLIA SONO
UGUALI A 4.0225 KM.
```

CAPITOLO SEI

Decisioni, ramificazioni e applicazioni

ARGOMENTI

La potenza del computer sta in gran parte nella sua capacità di prendere decisioni durante l'esecuzione di un programma. Nel presente capitolo discuteremo questa caratteristica e proseguiremo nell'apprendimento della programmazione in Basic.

Prendere decisioni nei programmi

In base a certe decisioni prese durante l'esecuzione di un programma, può succedere che il calcolatore salti a numeri di linea non in ordine numerico. Un tale trasferimento ad una linea di programma, può essere incondizionato, oppure può dipendere dai valori delle variabili relative a quel programma. L'uso di queste istruzioni di salto condizionato o incondizionato fa sì che anche programmi semplici diano risultati notevoli.

Applicazioni

Come nel precedente capitolo, vedremo come applicare le tecniche che studiamo ai programmi in Basic.

La ricerca degli errori nei programmi

Quasi tutti i programmi, nella loro prima versione, contengono errori. La ricerca e la correzione degli errori sono importantissime, e, come la programmazione stessa, possono essere imparate.

PRATICA SUL CALCOLATORE

Passiamo direttamente a lavorare sul calcolatore.

1. Selezionate il Basic sul vostro computer ed inserite il seguente programma:

```
100 LET X=1
110 PRINT X
120 LET X=X+1
130 IF X<5 THEN 110
140 END
```

   Il simbolo "<" alla linea 130 significa "minore di"; quindi, l'istruzione vuol dire "Se X è minore di 5 allora va a 110". Studiate attentamente il programma. Che cosa pensate sarà stampato se lo eseguite?

   ____

   Fate funzionare il programma e segnate quello che è successo.

   ____

2. Ora scrivete

```
100 LET X=2
```

   Listate il programma. Quale sarà l'output?

   ____

   Eseguite il programma e riportate i risultati.

   ____

3. Facciamo qualche altra modifica al programma per vedere se state seguendo quello che succede. Scrivete

```
120 LET X=X+2
```

Fate una lista del programma e studiatelo attentamente. Che cosa pensate farà ora?

Fate funzionare il programma e guardate se avete detto giusto. Copiate sotto quello che è effettivamente accaduto.

---

4. Riferendoci sempre al programma che avete in memoria, vogliamo fare un altro esperimento, ma questo richiede qualche cambiamento. Se volete, potete modificare il programma per renderlo uguale a quello riportato qui sotto, oppure potete cancellare il programma dalla memoria ed inserire quello nuovo.

```
100 LET X=1
110 PRINT X
120 LET X=X+1
130 IF X>=5 THEN 140
140 GOTO 110
150 END
```

Eseguite il programma e segnate quello che è accaduto.

Confrontate l'output annotato sopra con quello che avete copiato al passo 1. C'è qualche relazione?

---

5. Nel programma del punto 4, la linea 130 contiene un'asserzione. L'asserzione è X>= 5, che va letta "X è maggiore o uguale a 5". Se, per esempio, X avesse il valore numerico 6, l'asserzione sarebbe vera. Se X avesse il valore 3, l'asserzione sarebbe falsa. Consideriamo ora il programma del passo 4. Se esso viene eseguito, il computer parte dalla linea 100 e poi va alle linee 110, 120 e 130. Se l'asserzione della linea 130 è vera, a che numero di linea andrà poi il computer?

---

6. Finora nei programmi sono state usate soltanto due condizioni. Esse sono "<" (minore di) e ">=" (maggiore o uguale di). Come scrivereste la condizione "maggiore di"?

E come scrivereste "minore o uguale di"?

E "uguale a"?

Ed infine, come scrivereste "diverso da"?

Se riuscite a rispondere a queste domande senza troppa difficoltà, molto bene. Se non ci riuscite, non vi preoccupate, rivedremo tutto più avanti. Quello che importa adesso è come lavora l'istruzione IF...THEN.

7. Ora vedremo qualche applicazione nella quale viene usata l'istruzione IF...THEN. Eliminate il programma dalla memoria ed inserite il seguente:

```
100 PRINT "INSERITE 1,2 O 3";
110 INPUT Y
120 IF Y=1 THEN 150
130 IF Y=2 THEN 170
140 IF Y=3 THEN 190 ELSE 100
150 PRINT "SANGUE"
160 GOTO 100
170 PRINT "SUDORE"
180 GOTO 100
190 PRINT "LACRIME"
200 GOTO 100
210 END
```

Listate il programma e controllate se lo avete inserito correttamente. Studiatelo brevemente. Ricordate che quando il programma viene eseguito ed il computer stampa il segnale di input, si suppone che voi inseriate 1, 2 o 3. Che valore, o valori, di Y faranno sì che il calcolatore raggiunga la linea 120 del programma?

Che valore, o valori, di Y faranno raggiungere al calcolatore la linea 130?

Stessa domanda per la linea 140.

---

8. Supponete di voler ottenere la stampa di SUDORE. Che valore di Y dovrete inserire?

Guardate se avete detto giusto. Fate funzionare il programma ed inserite il numero che avete scritto. Che cosa è successo?

---

9. Quale valore di Y causerà la stampa di SANGUE?

E per far stampare al computer la parola LACRIME?

Controllate ognuna delle risposte che avete dato per vedere se avete detto giusto.

10. Il programma suppone che, in risposta al segnale di input, venga inserito 1, 2 o 3. Considerate il programma per un attimo e cercate di capire che cosa succederà se, al segnale di input, rispondete 4. Che cosa pensate che accadrà?

Eseguite il programma, inserite 4 ed annotate quello che è successo.

Si può spiegare facilmente quello che è accaduto nel programma, considerando cosa fa il computer quando incontra un'asserzione nell'istruzione IF...THEN. Ricordate, se l'asserzione è vera, il computer va al numero di linea che segue THEN. Se la condizione è falsa, il calcolatore passa al numero di linea successivo. Ora fate uscire il computer dal ciclo di input.

11. Cancellate lo schermo ed eliminate il programma dalla memoria; inserite ciò che segue:

```
100 A$="NERO"
110 B$="BIANCO"
120 C$="GATTO"
```

```
130 D$="CANE"
140 INPUT X
150 ON X GOTO 160,180,200,220
160 PRINT C$
170 GOTO 140
180 PRINT D$
190 GOTO 140
200 PRINT A$&C$
210 GOTO 140
220 PRINT B$&D$
230 GOTO 140
240 END
```

Il programma ha qualche caratteristica nuova. Primo, notate che vengono usate variabili di stringa introdotte nel capitolo 3. Esse sono definite nelle linee 100, 110, 120 e 130. Studiate il programma per qualche momento cercando di capire quello che fa. Ora proviamolo. Eseguitelo e, al segnale di input, scrivete 1. Che cosa è successo?

---

12. Il programma sta aspettando un altro input. Inserite 2. Che cosa è successo?

Questa volta, provate il numero 3.

Inserite 4 ed annotate quello che è successo.

---

13. Ora dovrebbe essere chiaro che il programma, alla linea 150, salta a numeri di linea differenti in base al valore di X. Nell'istruzione 130 ci sono quattro numeri di linea e abbiamo provato X = 1, 2, 3 o 4. Che cosa pensate accadrà se inseriamo 10?

Provate a farlo e segnate sotto quello che è successo.

Si spera che ormai abbiate compreso che cosa sta accadendo. Se non è così, non vi preoccupate, rivedremo tutto nella discussione.

14. Un ultimo programma e avremo finito con la parte pratica. Uscite

dal ciclo di input, cancellate il programma dalla memoria e inserite il seguente:

```
100 INPUT A$
110 INPUT B$
120 IF A$<B$ THEN 160
130 PRINT B$,A$
140 PRINT
150 GOTO 100
160 PRINT A$,B$
170 PRINT
180 GOTO 100
190 END
```

È chiaro che in questo programma il computer, in risposta ai segnali di input aspetterà l'inserimento di stringhe di caratteri. L'idea nuova ed interessante di questo programma sta nella linea 120. Guardatela attentamente. Che cosa pensate significhi il simbolo di "minore di" riguardo alle stringhe?

---

15. Vediamo ora come funziona il programma. Se lo eseguite e, al primo segnale di input, scrivete GATTO, e al secondo scrivete CANE, che cosa pensate farà il computer?

Provate a farlo e riportate quello che è successo.

---

16. Tutto bene, il computer è tornato indietro e sta aspettando ancora un input. Questa volta, inserite le parole MELA e ARANCIA. Che cosa è accaduto?

Ora provate con AARDVARK e ARK. Segnate che cosa ha stampato il computer.

Questo esercizio apre le porte ad alcune applicazioni non numeriche molto interessanti.

17. Fate uscire il computer dal ciclo di input. Questo conclude la parte pratica per ora. Scrivete BYE, spegnete il computer e passate alla discussione.

DISCUSSIONE

In questo capitolo avremo a che fare con due argomenti. Il primo è il concetto di istruzione di salto, sia condizionato che incondizionato, e l'uso di queste istruzioni nei programmi. Il secondo riguarda la capacità di trovare e correggere gli errori di un programma.

Salto incondizionato

Dall'inizio di questo libro, abbiamo continuamente usato istruzioni di salto incondizionato. Il seguente programma illustra l'uso di questa istruzione.

```
100 LET Z=2
110 PRINT Z
120 LET Z=2*Z
130 GOTO 110
140 END
```

Ricordate che quando fate funzionare un programma, il computer va all'istruzione con il numero di linea più basso e poi esegue le istruzioni in ordine di numero di linea crescente. Il solo modo di interrompere questo percorso è con un'istruzione di salto (oppure, come vedremo nel prossimo capitolo, con un comando di ciclo). Nel programma sopra, il calcolatore eseguirebbe i numeri di linea come segue: 100, 110, 120, 130, 110, 120, 130, e così via. Il punto è che l'istruzione 130 fa saltare il computer indietro alla linea 110, invece di andare alla 140. Notate che non ci sono condizioni nell'istruzione della linea 130. Questo è il motivo per cui l'istruzione GOTO è nota come istruzione di salto "incondizionato". È anche chiaro che l'istruzione GOTO mette il programma in un ciclo dal quale non si può uscire. Il solo modo di far uscire il computer da questo ciclo, è di interrompere il programma dalla tastiera mentre sta funzionando.
Per riassumere, se in qualche punto di un programma volete che il computer salti ad un'altra linea senza condizioni, usate l'istruzione GOTO. State però attenti a non "intrappolare" il programma in un ciclo.

Salto condizionato

Molto probabilmente, avremo ormai stabilito una connessione tra le istruzioni IF...THEN, che avete incontrato nella parte pratica, e la nozione di salto "condizionato". Tutte le istruzioni di salto condizionato han-

no la stessa forma. Ecco una descrizione della loro struttura ed un esempio di istruzione IF...THEN:

Numero di linea IF<(relazione)> <(condizione)> <(relazione)>THEN Numero di linea

```
240 IF 3*X-2>Y-Z THEN 360
```

Tutte le istruzioni IF...THEN hanno questo formato. IF e THEN, così come i due numeri di linea dell'istruzione, non richiedono spiegazioni particolari. Il cuore dell'istruzione sta nelle due espressioni separate dal simbolo di condizione, che formano l'asserzione. Le considereremo molto attentamente.
In tutti gli esempi visti finora, ad eccezione di quello sopra, le relazioni erano fra variabili numeriche, variabili stringa o costanti. Questo è il tipo di asserzione usata più spesso nei programmi. Degli esempi potrebbero essere

```
100 IF U<3 THEN 250
340 IF S$>T$ THEN 220
```

Ci sono tuttavia dei casi in cui si potrebbero voler usare delle espressioni più complicate. Nell'esempio che seguiva la descrizione dell'istruzione IF...THEN, la prima relazione era

```
3*X-2
```

dove è bene che ad X sia stato assegnato un valore. Anche la seconda relazione

```
Y-Z
```

può essere usata se Y e Z hanno dei valori. Per illustrare ulteriormente quello che succede in un programma, supponiamo che X abbia il valore 1, Y sia 10 e Z sia 4. Il calcolatore tradurrà l'istruzione

```
240 IF 3*X-2>Y-Z THEN 360
```

in questo modo. Come prima cosa, sostituira i valori di X, Y e Z, il che cambierà l'istruzione in

```
200 IF 1>6 THEN 360
```

Prima o poi, tutte le istruzioni IF...THEN che implicano variabili nume-

riche vengono ridotte a questa forma, in cui il computer deve giudicare se un'asserzione, stabilita da due numeri ed una condizione, è vera o falsa. Se sono coinvolte variabili stringa, il confronto viene fatto diversamente, come sarà spiegato più avanti. In questo caso l'asserzione 1>6 è falsa. Tuttavia un'asserzione come 4 <10 sarebbe vera. Se l'asserzione è vera, il calcolatore passerà al numero di linea che segue THEN. Se invece è falsa, il computer andrà al prossimo numero di linea del programma. Se lo desideriamo, possiamo usare una diversa versione di IF...THEN. Un esempio di questa nuova istruzione è

```
300 IF X>Y THEN 240 ELSE 43
```

Se l'asserzione è vera, il programma salterà alla linea 240; se è falsa passerà alla 435.
Nelle istruzioni IF...THEN si possono usare parecchie condizioni. Eccone una lista con il loro significato.

Condizione	Significato
=	Uguale a
<	Minore di
>	Maggiore di
<=	Minore o uguale di
>=	Maggiore o uguale di
<>	Diverso da

Istruzioni di salto multiplo

Nella parte pratica abbiamo visto che è possibile, usando una sola istruzione, saltare a parecchi punti differenti di un programma. Utilizziamo il seguente programma per vedere come questo viene fatto.

```
200 ON A GOTO 310,320,330
210 B=A+2
```

Nella linea 200 la decisione concernente la linea a cui saltare è basata sul valore di A. Se, per esempio, A fosse 1, il programma proseguirebbe al primo numero di linea della lista, che in questo caso sarebbe la 310. Similmente, se A fosse 3, il programma salterebbe alla linea 330, il terzo numero della lista. Nell'esempio riportato sopra, A poteva essere 1, 2 o 3, poiché ci sono tre numeri di linea nella lista di indirizzi.
Potreste chiedervi che cosa sarebbe successo se A avesse avuto qualche

altro valore, diciamo 8. La risposta è che, quando il computer non è in grado di localizzare un appropriato numero di linea nella lista di indirizzi, scrive *BAD VALUE IN 200 e si ferma. I numeri di linea nella lista di indirizzi che segue ON...GOTO non devono essere in ordine particolare. Per di più, se desiderate, potete ripetere lo stesso numero di linea nella lista. Se pensate un po' a questo potete immaginare le possibilità implicite.
La possibilità di controllare questo processo di salto cambiando il valore di una variabile numerica è il cuore dell'istruzione ON...GOTO. Questa istruzione di salto multiplo fornisce un mezzo di scelta molto potente che ha parecchie applicazioni nei programmi in Basic.

Condizioni non numeriche di salto

Come abbiamo visto, è possibile usare variabili stringa nelle istruzioni IF...THEN. Il confronto tra stringhe è basato sulla posizione alfabetica. Così, A è minore di B, perché A, nell'alfabeto, viene prima di B. Similmente, Z è maggiore di T, perché compare dopo T.
Possiamo estendere questa idea alle parole, nel qual caso il confronto è fatto carattere per carattere. Per esempio, CERO è maggiore di CERA. I primi tre caratteri in entrambi le parole sono uguali, quindi in questa parte delle stringhe non viene riscontrata nessuna differenza. Tuttavia il quarto carattere O viene dopo di A, così CERO è considerato maggiore di CERA. Nel caso di stringhe di lunghezza diversa, il confronto viene fatto finché possibile, limitato dalla lunghezza della stringa più corta. Così, CERO è minore di CEROTTO. Il confronto indica uguaglianza per i primi quattro caratteri (la lunghezza della stringa più corta), ma ci sono degli altri caratteri in CEROTTO, da cui la valutazione. Naturalmente, CORO è considerato maggiore di CEROTTO.
Una volta capito il concetto di confronto tra caratteri, si possono usare variabili stringa in istruzioni di salto condizionato nello stesso modo delle variabili numeriche. Dovrebbe essere chiaro che questa capacità di comparare stringhe è veramente potente e rende molto semplice l'ordinamento alfabetico di una lista di parole. Di questo vedremo parecchi esempi più avanti.

ESEMPI DI PROGRAMMI

Fino a questo punto i nostri programmi avevano dei grossi difetti. Da un lato, il programma poteva implicare la ripetizione di un processo, ma non c'era mòdo di fermare quest'ultimo. Alle volte, il programma veni-

va fermato, ma era spesso banale. Quello che vogliamo, è un modo di ottenere un programma che svolga un compito utile (il che può implicare la ripetizione di certi procedimenti) e che poi si fermi automaticamente. Un sistema per farlo ci è dato dalle istruzioni di salto condizionato che abbiamo appena imparato. Vedremo adesso parecchi programmi che illustrano questa possibilità.

Esempio 1 - Stampa di tabelle di numeri

Il nostro problema è scrivere un programma che, quando eseguito, stampi la seguente tabella di numeri:

```
2    3
4    5
6    7
8    9
```

Ci sono parecchie caratteristiche di questa tabella che vanno considerate quando si scrive il programma. La prima è che il 2 e i numeri successivi sono staccati mediante la spaziatura standard (due numeri per linea). Ogni numero è maggiore del precedente di 1. L'ultimo numero stampato è 9; poi il calcolatore dovrebbe fermarsi.
Sono possibili parecchie soluzioni. Un programma, che non è il migliore, ma funzionerebbe, è

```
100 PRINT 2,3,4,5,6,7,8,9
110 END
```

Potreste controllare questo programma per accertarvi che produce effettivamente la tabella voluta. Esso illustra anche un concetto molto importante. In effetti, non esiste "il" programma esatto. Il solo controllo che si può fare è "il programma funziona correttamente?". Certamente alcuni programmi sono più geniali, oppure possono raggiungere i risultati più efficacemente di altri, ma questa è un'altra questione. Il principiante dovrebbe solo preoccuparsi del fatto che un programma fornisca o no i risultati desiderati, e non di questioni di stile.
Ma ritorniamo al nostro problema. Un possibile approccio consiste nel far stampare al computer il primo numero della tabella. Vogliamo anche organizzare il programma in modo che richieda un'unica istruzione di stampa. Questo implica che venga stampato il valore di una variabile, che sarà cambiato durante l'esecuzione. Possiamo iniziare il nostro programma con la seguente parte:

```
100 LET X=2
110 PRINT X,
```

Ad X viene dato il valore 2, che viene stampato alla linea 110. La virgola fa sì che il computer si sposti alla prossima posizione standard di stampa. Ora dobbiamo generare il successivo valore da stampare. Notate che, in qualsiasi punto della tabella, il prossimo numero è proprio 1 in più di quello attuale. Questo può essere fatto con

```
120 LET X=X+1
```

Ora tutto quello che resta da fare è decidere se ritornare o no all'istruzione PRINT. Finché X è minore o uguale a 9, torneremo indietro. Questo può essere fatto con un'istruzione di salto condizionato.

```
130 IF X<=9 THEN 110
```

Alla fine mettiamo l'istruzione END. Il programma completo è

```
100 LET X=2
110 PRINT X,
120 LET X=X+1
130 IF X<=9 THEN 110
140 END
```

Questo è un programma semplice ed ha scarso valore pratico, ma serve ad illustrare come si utilizzano le istruzioni di salto condizionato per uscire dal programma al momento giusto.

Esempio 2 - Bollo automobilistico

Supponiamo che, nel tentativo di forzare i guidatori ad usare automobili che sviluppano pochi hp, e quindi a risparmiare energia, lo stato stabilisca che il prezzo del bollo dipenda dalla potenza della macchina. Riportiamo il criterio seguito ed il prezzo del bollo a seconda della categoria a cui appartiene l'automobile.

Potenza in hp	Prezzo del bollo
Fino a 50 hp	0
Più di 50 hp, fino a 100 hp	30 000
Più di 100 hp, fino a 200 hp	70 000
Più di 200 hp, fino a 300 hp	150 000
Più di 300 hp	500 000

Vogliamo un programma che, quando eseguito, fornisca il seguente tipico output.

```
QUANTI HP ? 325
IL PREZZO DEL BOLLO E' 500000

QUANTI HP ? 85
IL PREZZO DEL BOLLO E' 30000

(ECC.)
```

Chiaramente, il solo punto difficile del programma, sarà decidere il prezzo del bollo. Questo processo di decisione servirà a chiarire l'uso dell'istruzione IF...THEN. Per iniziare, scriveremo le istruzioni per l'input della potenza in hp. La potenza dell'automobile sarà indicata con P. Seguite il formarsi del programma ma non cominciate a scriverlo finché non sarà completo. Inizialmente trascriveremo alcune parti del programma per poi riprenderle e completarle nei dettagli.
Se cominciate a scriverlo con le parti mancanti il computer segnalerà degli errori. Il programma può iniziare con:

```
100 PRINT "QUANTI HP";
110 INPUT P
```

Ora dobbiamo elaborare un metodo per decidere in che categoria sta P. Un modo logico di farlo, sarebbe di controllare dalla più bassa categoria in su. All'inizio possiamo verificare se P è minore o uguale a 50. Se è così, allora sappiamo che non c'è nulla da pagare.

```
120 IF P<=50 THEN    (NON OCCORRE BOLLO)
```

Il numero di linea che dovrebbe seguire THEN manca per una ragione ben precisa. Se il numero in P è minore o uguale a 50, vogliamo che il calcolatore salti ad un'istruzione che assegnerà alla variabile F, rappresentante il prezzo del bollo, il valore 0. Il problema è che, a questo punto, non sappiamo che numero di linea sarà usato per questa istruzione. Conseguentemente, lo lasceremo vuoto e, più avanti, vi inseriremo il valore corretto. La nota dopo il numero di linea mancante, è stata messa per ricordarci che, se l'asserzione della linea è vera, non occorre il bollo. Se l'asserzione della linea 120 è falsa, il computer andrà al successivo numero di linea. In questo caso vogliamo vedere se P appartiene alla prossima categoria.

```
130 IF P<=100 THEN   (PREZZO DEL BOLLO  30000)
```

Anche in questo caso, non sappiamo che numero di linea mettere dopo THEN, ma lo possiamo fissare successivamente. Per stabilire completamente a che categoria appartiene P, ci sono ancora tre istruzioni di salto. Ora che si è capito come fare, le possiamo scrivere tutte assieme.

```
140 IF P<=200 THEN   (PREZZO DEL BOLLO  70000)
150 IF P<=300 THEN   (PREZZO DEL BOLLO 150000)
160 IF P>300 THEN    (PREZZO DEL BOLLO 500000)
```

A questo punto il programma è

```
100 PRINT "QUANTI HP";
110 INPUT P
120 IF P<=50 THEN    (PREZZO PEL BOLLO      0)
130 IF P<=100 THEN   (PREZZO DEL BOLLO  30000)
140 IF P<=200 THEN   (PREZZO DEL BOLLO  70000)
150 IF P<=300 THEN   (PREZZO DEL BOLLO 150000)
160 IF P>300 THEN    (PREZZO DEL BOLLO 500000)
```

Ora possiamo fissare il numero di linea mancante alla linea 120. Poiché il prossimo numero di linea del programma sarebbe 170, possiamo usare proprio questo.

```
100 PRINT "QUANTI HP";
110 INPUT P
120 IF P<=50 THEN 170
130 IF P<=100 THEN   (PREZZO DEL BOLLO  30000)
140 IF P<=200 THEN   (PREZZO DEL BOLLO  70000)
150 IF P<=300 THEN   (PREZZO DEL BOLLO 150000)
160 IF P>300 THEN    (PREZZO DEL BOLLO 500000)
170 LET F=0
180 GOTO    (istruzione PRINT)
```

Anche adesso, nella linea 180, manca un numero di linea. La nota indica che vogliamo passare ad un'istruzione PRINT. Se l'asserzione nella linea 120 è vera, il computer salta alla linea 170 ed assegna a F il valore 0.
Proseguendo allo stesso modo, possiamo inserire i numeri mancanti alle linee 130, 140, 150 e 160. Il risultato è

```
100 PRINT "QUANTI HP";
110 INPUT P
120 IF P<=50 THEN 170
130 IF P<=100 THEN 190
140 IF P<=200 THEN 210
150 IF P<=300 THEN 230
160 IF P>300 THEN 250
170 LET F=0
180 GOTO        (istruzione PRINT)
190 LET F=30
200 GOTO        (istruzione PRINT)
210 LET F=70
220 GOTO        (istruzione PRINT)
230 LET F=150
240 GOTO        (istruzione PRINT)
250 LET F=500
```

La prossima linea del programma sarebbe la 260; possiamo usarla per l'istruzione PRINT. Il resto del programma segue facilmente. Ecco il programma completo.

```
100 PRINT "QUANTI HP";
110 INPUT P
```

```
120 IF P<=50 THEN 170
130 IF P<=100 THEN 190
140 IF P<=200 THEN 210
150 IF P<=300 THEN 230
160 IF P>300 THEN 250
170 LET F=0
180 GOTO 260
190 LET F=30
200 GOTO 260
210 LET F=70
220 GOTO 260
230 LET F=150
240 GOTO 260
250 LET F=500
260 PRINT "IL PREZZO DEL BOLLO E' ";
F*1000
270 PRINT
280 GOTO 100
290 END
```

Ora che abbiamo aggiunto tutti i numeri di linea mancanti possiamo inserire il programma nel computer e verificare che esso lavora in modo corretto.
Può darsi abbiate notato che l'istruzione di salto condizionato della linea 160 non è necessaria. Per capire il perché, considerate le asserzioni nelle istruzioni IF...THEN. Se l'asserzione della linea 120 è falsa, sappiamo che P deve essere maggiore di 50. Similmente, se ognuna delle successive asserzioni è falsa, il calcolatore passa alla linea seguente. In particolare, supponiamo che il computer raggiunga la linea 150 e determini che l'asserzione è falsa. Questo indirizza il calcolatore alla linea 160, ma allora sappiamo che P deve essere maggiore di 300 e quindi si può stampare il prezzo del bollo, senza ulteriori confronti. Se assegniamo ad F il valore 5000000 nella linea 160, ne risulta un programma leggermente diverso:

```
100 PRINT "QUANTI HP";
110 INPUT P
120 IF P<=50 THEN 200
130 IF P<=100 THEN 220
140 IF P<=200 THEN 240
150 IF P<=300 THEN 260
160 LET F=500
170 PRINT "IL PREZZO DEL BOLLO E' ";
F*1000
180 PRINT
190 GOTO 100
200 LET F=0
210 GOTO 170
220 LET F=30
230 GOTO 170
240 LET F=70
250 GOTO 170
260 LET F=150
270 GOTO 170
280 END
```

Entrambe le versioni del programma lavoreranno ugualmente bene e, se volete, potete scriverne una vostra. Il modo di fissare le ramificazioni di un programma è una questione che dovrete decidere voi. La sola cosa di cui vi dovrete preoccupare è se il vostro programma funziona o no.

Abbiamo trattato questo programma nei dettagli perché spesso il principiante trova difficoltà nello scrivere programmi che implicano tali regole di ricerca. Dovreste studiare il programma finché siete convinti che esso fa esattamente quello che desiderate. Cercate anche di ricordarvi di usare la tecnica di tralasciare i numeri di linea quando non sapete quali dovranno essere, e ritornate dopo ad inserire i valori adatti. In questi casi, i commenti alla destra vi aiuteranno a ricordare che cosa volete succeda a quel punto del programma. Tuttavia ricordate anche, se tralasciate di scrivere i numeri di linea mentre scrivete il programma, di non tentare di introdurre le linee nel computer finché il programma non è completo.

Esempio 3 - Calcolo della media

Supponiamo di avere due numeri in un'istruzione DATA, e di volerne calcolare la media. Il problema è che non sappiamo in anticipo quanti numeri ci sono. Così useremo una "variabile flag" per individuare la fine dei dati.
Il flag sarà un numero che potrà comparire tra i dati molto difficilmente. Nel nostro caso useremo il numero 9999, ma voi potreste sceglierne un altro a vostro piacere.
Ecco come funziona il flag. L'istruzione DATA apparirà sempre come segue:

Numero di linea   DATA (numero), (numero),...,(numero),9999

Il flag 9999 è posto insieme ai dati, dopo l'ultimo numero da usare per calcolare la media. Nel programma, ogni volta che leggiamo un numero dall'istruzione DATA, dobbiamo controllarlo per vedere se è 9999. Se non lo è, sappiamo che questo numero deve essere incluso nel calcolo della media. Se il numero è 9999 vuol dire che abbiamo già letto tutti i dati e possiamo passare al resto del programma.
La media viene calcolata dividendo la somma dei numeri per il numero dei numeri. Nel nostro programma, dobbiamo calcolare entrambe queste quantità. Useremo S per indicare la somma dei numeri ed N per il loro numero. Al momento dell'esecuzione del programma, non sappiamo quali sono questi valori, quindi li dobbiamo porre uguali a 0 e poi aggiornarli man mano che si leggono i numeri dell'istruzione DATA.
Il programma comincia fissando i valori iniziali di S e di N.

```
100 LET S=0
110 LET N=0
```

In effetti non occorreva che lo facessimo, dato che il computer pone automaticamente a zero le variabili numeriche. Tuttavia, se queste istruzio-

ni ci sono, il programma è più chiaro. Ora possiamo leggere un numero dall'istruzione DATA e confrontarlo con il valore del flag.

```
120 READ X
130 IF X=9999 THEN      (calcolo della media)
```

Stiamo usando il metodo, introdotto prima, di non scrivere un numero di linea in un'istruzione di salto condizionato finché non sappiamo quale dovrà essere. In questo caso, se l'asserzione (X = 9999) è vera, sappiamo che tutti i numeri dell'istruzione DATA sono stati considerati e siamo pronti per il calcolo della media. Se l'asserzione è falsa, allora il numero appena letto deve far parte dei dati e deve essere valutato. Questo viene fatto come segue:

```
140 LET S=S+X
150 LET N=N+1
```

Nella linea 140, il valore di X (il numero appena letto) è sommato al valore di S. Ricordate che la somma di tutti i numeri, di cui poi calcoleremo la media, si sta accumulando in S. Nella linea 150, il numero N è incrementato di 1 per indicare che è stato considerato un altro numero. Dopo aver trattato il valore di X saltiamo indietro all'istruzione READ per continaure il procedimento.

```
160 GOTO 120
```

Ora, dato che il prossimo numero di linea del programma sarebbe probabilmente 170, possiamo mettere il numero mancante nella linea 130. Nella linea 170 viene calcolata la media, che sarà identificata con A. Se includiamo una tipica istruzione DATA, il programma completo è

```
100 LET S=0
110 LET N=0
120 READ X
130 IF X=9999 THEN 170
140 LET S=S+X
150 LET N=N+1
160 GOTO 120
170 LET A=S/N
180 PRINT A
190 DATA 4,2,3,6,5,9999
200 END
```

Naturalmente, possiamo avere tante istruzioni DATA quante sono necessarie per contenere tutti i numeri di cui vogliamo calcolare la media. Dopo l'ultimo numero dell'ultima istruzione DATA, ci sarà il flag 9999 per indicare la fine dei dati. Questo ci farà uscire dal ciclo di lettura e ci indicherà quando calcolare la media. L'istruzione di salto condizionato, assieme all'idea di variabile flag, ci dà un potente strumento da utilizzare nei programmi.

Esempio 4 - Pagamento dell'interesse su un prestito

L'ammontare dell'interesse su un prestito dipende dalla somma richiesta. Supponiamo che una banca abbia la normativa seguente: 5% per i primi 25 milioni, 10% per i prossimi 10 milioni, 15% di quello che resta fino a 50 milioni; il prestito non è concesso se la cifra supera i 50 milioni. Il nostro problema è di scrivere un programma che richieda in input l'ammontare del prestito richiesto, poi calcoli e stampi l'ammontare dell'interesse. Se la somma richiesta supera i 50 milioni verrà stampato un messaggio indicante che il prestito non è concesso.
Prima, mettiamo le istruzioni per l'input della cifra richiesta in prestito.

```
100 PRINT "PRESTITO RICHIESTO";
110 INPUT P
```

Ora controlliamo se P è inferiore del limite.

```
120 IF P<=50000000 THEN 150
130 PRINT "PRESTITO NON CONCESSO"
140 GOTO          (istruzione END)
```

Per il momento non scriviamo il numero di linea nella riga 140, dato che non sappiamo ancora quale sarà il numero di linea dell'istruzione END. Il commento sulla destra serve a ricordarci dove si deve saltare con quell'istruzione.
Poi controlleremo se P è maggiore o uguale a 35 milioni, oppure maggiore o uguale a 25 milioni. A seconda delle risposte, possiamo calcolare l'ammontare dell'interesse.

```
150 IF P>=35000000 THEN      (?)
160 IF P>=25000000 THEN      (?)
170 LET D=.05*P
180 GOTO        (istruzione PRINT)
```

Osservate che se le asserzioni nelle linee 150 e 160 sono false, sappiamo che P è minore di 25 milioni e possiamo calcolare l'interesse nella linea 170. Il posto vuoto nella linea 180 sarà, quando lo sapremo, il numero di linea dell'istruzione PRINT finale. Poiché la prossima linea sarebbe 190, possiamo usare questo numero come indirizzo a cui saltare dalla linea 150.

```
150 IF P>=35000000 THEN 190
160 IF P>=25000000 THEN      (?)
170 LET D=.05*P
180 GOTO       (istruzione PRINT)
190 LET D=.05*25000000+.1*10000000
+.15*(P-35000000)
200 GOTO       (istruzione PRINT)
```

Ora possiamo usare il numero di linea 210 per il salto dalla linea 160.

```
150 IF P>=35000000 THEN 190
160 IF P>=25000000 THEN 210
170 LET D=.05*P
180 GOTO      (istruzione PRINT)
190 LET D=.05*25000000+.1*10000000
+.15*(P-35000000)
200 GOTO      (istruzione PRINT)
210 LET D=.05*25000000+.1*(P-25
000000)
```

L'istruzione **PRINT** può andare alla linea 220, seguita dall'istruzione **END**.

```
220 PRINT "L'INTERESSE E'";D
230 END
```

Ora sappiamo che l'istruzione **PRINT** è alla linea 220 e l'istruzione **END** alla linea 230. Mettendo questi numeri negli spazi vuoti a loro riservati, abbiamo completato il programma.

```
100 PRINT "PRESTITO RICHIESTO";
110 INPUT P
120 IF P<=50000000 THEN 150
130 PRINT "PRESTITO NON CONCESSO"
140 GOTO 230
150 IF P>=35000000 THEN 190
160 IF P>=25000000 THEN 210
170 LET D=.05*P
180 GOTO 220
190 LET D=.05*25000000+.1*10000000
+.15*(P-35000000)
200 GOTO 220
210 LET D=.05*25000000+.1*(P-25
000000)
220 PRINT "L'INTERESSE E'";D
230 END
```

RICERCA DEGLI ERRORI NEI PROGRAMMI

La capacità di esaminare un programma e stabilire se esso funzionerà o no, è certamente una delle cose più importanti che un principiante deve imparare. Più precisamente, quando un programma non funziona a dovere, siete in grado di trovare gli errori e correggerli? La correzione dei programmi sembra inizialmente molto difficile. Tuttavia, una volta imparato, il programmatore ha solitamente difficoltà a capire come mai gli altri non ci riescano.
La correzione dei programmi richiede due cose. Primo, dovete essere in grado di capire che significato ha un'istruzione Basic per il computer. Poi dovete riuscire a seguire un programma, controllando ogni passo ed azione nei dettagli, supponendo che vengano eseguiti. Ora che conoscia-

mo abbastanza il Basic, possiamo dedicare un po' di tempo alla correzione dei programmi. Il tempo trascorso su questa parte vi farà risparmiare molto più tempo successivamente.

Traduzione delle istruzioni Basic

Abbiamo usato vari tipi differenti di istruzioni. Vogliamo vedere che cosa fa il computer quando le esegue. Come esempio, supponiamo che il calcolatore valuti l'istruzione

```
140 LET X=3
```

Questa istruzione dice al calcolatore di fissare una locazione di memoria, chiamarla X, e di memorizzare 3 in quella locazione. Similmente

```
160 LET B=0
```

indica al computer di chiamare una locazione B e di memorizzarvi 0. Con la seguente istruzione, la situazione è un po' più complicata:

```
135 LET X=A+B-2
```

Ora il calcolatore è indirizzato a prendere i numeri memorizzati in A e B, fare la somma, sottrarre 2, e porre il risultato in una locazione chiamata X. Tutto questo è giusto ammesso che il computer possa trovare le locazioni chiamate A e B. Se non sono state fissate prima dell'esecuzione dell'istruzione, il calcolatore le cercherà, e non trovandole, le fisserà ponendo zero in entrambe, dopo di che proseguirà nell'esecuzione. Naturalmente, questo potrebbe non essere affatto quello che volevamo, quindi bisogna stare un po' attenti.
Che cosa succede quando il calcolatore incontra un'istruzione del tipo

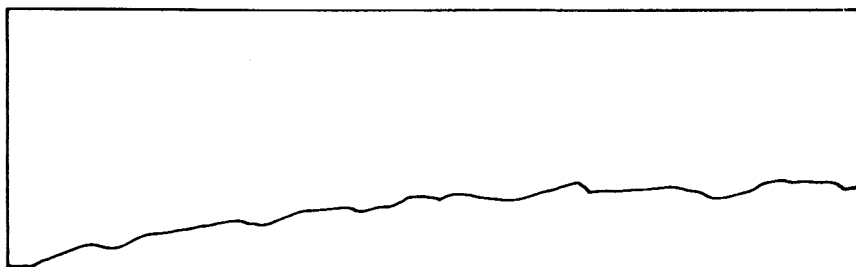
```
185 IF M=N THEN 240
```

che dice di vedere se i numeri contenuti in M ed N sono uguali? Se i numeri sono uguali, la prima linea da eseguire sarebbe la 240. Altrimenti, il calcolatore passerà alla successiva linea in ordine numerico. Se il computer non trova le locazioni M e N, le fisserà ponendovi degli zeri. Questo assicura che l'asserzione sarà vera. Anche in questo caso, bisogna stare attenti che tutte le variabili siano fissate come richiesto dal problema, altrimenti potrebbero accadere strane cose.
Vogliamo ora utilizzare queste nozioni per localizzare gli errori che ci possono essere in un programma.

Tracciamento dei programmi

Il programma sviluppato nell'Esempio 3 del precedente paragrafo, costituisce un buon esempio da usare per capire come si effettua la correzione dei programmi. Il programma viene ripetuto come riferimento.

```
100 LET S=0
110 LET N=0
120 READ X
130 IF X=9999 THEN 170
140 LET S=S+X
150 LET N=N+1
160 GOTO 120
170 LET A=S/N
180 PRINT A
190 DATA 4,2,3,6,5,9999
200 END
```

Quello che vogliamo fare ora, è convincervi che il migliore aiuto alla programmazione è un semplice foglio di carta! Usato correttamente, questo incredibile sussidio alla programmazione vi permetterà di trovare tutti gli errori dei vostri programmi e vi indicherà come correggerli. Il compito può sembrare enorme per un dispositivo così semplice, ma è così. Più avanti vedremo alcune funzioni particolari del TI home computer che vi aiuteranno nella correzione dei programmi ma per il momento vogliamo eseguirle a mano.
Prima ricopiate il programma su un foglio di carta a righe e seguite la nostra discussione usando questa copia. Coprite tutto il programma, tranne la prima riga, con un altro foglio di carta.

Ora consideriamo la prima istruzione, che dice al computer di fissare una locazione di memoria chiamata S, e di metterci zero. Useremo il nostro foglio per segnare il contenuto della memoria. Scriveremo quindi una S e sotto uno 0.

```
100 LET S=0
```

Questo è tutto per la prima linea del programma. Spostate leggermente il foglio in modo da vedere la prossima linea. Ricordatevi che state sostenendo la parte del computer e state utilizzando il foglio di carta sia per riportare il contenuto della memoria che per considerare soltanto una linea di programma alla volta.

```
110 LET N=0
```

Passiamo alla linea 120

```
120 READ X
```

Questa indica al calcolatore di leggere un numero dall'istruzione DATA, che in questo caso è 4. Il 4 viene memorizzato nella locazione chiamata X.
Fermiamoci un attimo per rivedere quello che stiamo facendo. Stiamo seguendo il programma linea per linea riportando quello che il calcolatore deve fare. Poiché dobbiamo ancora trovare istruzioni di salto, valutiamo semplicemente l'istruzione e poi passiamo alla prossima linea in ordine numerico. Ed ora andiamo alla riga 130.

```
130 IF X=9999 THEN 170
```

L'asserzione della riga 130 (X = 9999) viene calcolata usando il valore di X che appare sul foglio. Poiché a questo punto X ha valore 4, l'asserzione (4 = 9999) è falsa. Quindi, invece di andare alla riga 170, passiamo direttamente a quella seguente.

```
140 LET S=S+X
```

Prendiamo il numero che sta in S (0) e quello in X (4), li sommiamo e memorizziamo in S il risultato 4. Notate che in questo modo perdiamo il valore che S conteneva precedentemente. Incolonneremo i nuovi valori sotto quelli vecchi, per indicare che questi sono stati persi. In qualsiasi punto del programma, una variabile numerica avrà come valore l'ultimo numero della colonna corrispondente. Ora il computer esamina la riga 150.

```
150 LET N=N+1
```

S	N	X
Ø	Ø	4
4	1	

Con questa, a 0 viene aggiunto 1 ed il risultato diventa il nuovo valore di N, cosicché lo 0 viene perso. La linea 160 rimanda il computer all'istruzione READ della linea 120. Quindi tutto il procedimento viene ripetuto. Rimaniamo nel ciclo finché tutti i dati sono stati letti ed elaborati. Se seguite il programma finché nella variabile X è inserito il valore convenzionale 9999, sul vostro foglio avrete qualcosa di questo tipo:

```
130 IF X=9999 THEN 170
```

S	N	X
Ø	Ø	~~4~~
~~4~~	~~1~~	~~2~~
~~6~~	~~2~~	~~3~~
~~9~~	~~3~~	~~6~~
1~~5~~	~~4~~	~~5~~
20	5	9999

Poiché ora il valore di X è 9999, la condizione (X = 9999) è verificata, ed il computer salta alla linea 170.

```
170 LET A=S/N
```

Viene introdotta la nuova variabile A, che contiene il risultato della divisione tra il numero contenuto in S ed il valore di N. Infine il computer passa alla linea 180 per stampare il valore di A. L'elenco dei valori delle variabili ci prova che il calcolatore ha fatto ciò che volevamo ed ha dato i risultati corretti.
Esaminiamo ora un programma sbagliato ed usiamo questo metodo di controllo delle variabili per cercare l'errore. Supponiamo che il programma calcoli la somma di numeri inseriti dalla tastiera. Ogni volta che il computer stampa un segnale di input (punto interrogativo), introduciamo un numero. Quando tutti i valori sono stati inseriti, scriviamo 11111 per segnalare che i dati sono finiti. Il calcolatore dovrebbe stampare la somma dei numeri che precedono il segnale di fine flusso. Il seguente programma è sbagliato.

```
100 LET S=0
110 INPUT Y
120 IF Y=11111 THEN 150
130 LET S=S+Y
140 GOTO 100
150 PRINT S
160 END
```

Per scoprire cosa c'è di sbagliato, useremo il nostro metodo di controllo. Supponiamo che, all'apparire dei segnali di input, sia stata inserita la seguente serie di numeri:

3, 1, 6, 5, 11111

La somma dei valori che precedono il simbolo di fine flusso è 15, perciò sappiamo già cosa dovrà stampare il computer.

All'inizio il nostro foglio è bianco ed esaminiamo la prima riga del programma.

Allora

```
110 INPUT Y
```

S Y
0 3

Poiché Y non è 11111, passiamo alla linea 130,

```
130 LET S=S+Y
```

Dopo di che ritorniamo alla linea 100,

```
100 LET S=0
```

Se analizzate il programma finché Y assume il valore di 11111, alla fine sul vostro foglio avrete:

```
120 IF Y=11111 THEN 150
```

```
S     Y
Ø     3
3     1
Ø     6
1     5
Ø     11111
6
Ø
5
0
```

Siccome a questo punto Y ha valore 11111, il computer salta alla linea 150, che provvede a stampare il contenuto di S. Ma questo numero è 0, cioè è sbagliato. Se avete seguito attentamente il programma, passo per passo, probabilmente avete già scoperto cosa c'è che non va. L'errore è nell'istruzione di salto incondizionato della linea 140. Passando alla linea 100, il valore di S (che dovrebbe essere la somma dei numeri che sono stati inseriti) viene azzerato ogni volta che il numero è dato in input. Il problema è risolto facilmente modificando la linea 140 come segue

```
140 GOTO 110
```

Nel Basic del TI home computer sono state introdotte alcune funzioni che vi possono aiutare nella correzione di un programma per scoprire cosa c'è di sbagliato. Per illustrarle, torniamo al programma che calcola la media.

```
100 LET S=0
110 LET N=0
120 READ X
130 IF X=9999 THEN 170
140 LET S=S+X
150 LET N=N+1
160 GOTO 120
170 LET A=S/N
180 PRINT A
190 DATA 4,2,3,6,5,9999
200 END
```

Se prima di eseguire il programma scrivete TRACE, e poi date il RUN, il computer scriverà i numeri di linea man mano che il programma proce-

de. Tale comando, usato in questo programma farà apparire sullo schermo:

```
<100><110><120><130><140>
<150><160><120><130><140>
<150><160><120><130><140>
<150><160><120><130><140>
<150><160><120><130><140>
<150><160><120><130><170>
<180> 4
<200>
```

Se confrontate i numeri di linea stampati dal comando TRACE nel programma vedrete il cammino seguito dal computer.
In questo caso è accaduto che i vari TRACE relativi all'istruzione READ di riga 120 si ripetono periodicamente e si trovano incolonnati. Dopo sei READ (l'ultima è del dato 9999), c'è un salto alla linea 170. Dopo il 180 viene stampata la risposta che è 4. Notate che, poiché nel programma l'istruzione PRINT A non è seguita da punteggiatura, il seguente valore del TRACE viene stampato a capo riga. Dovrebbe essere chiaro che la possibilità di usare la funzione TRACE per seguire l'esecuzione di un programma dai numeri di linea è un mezzo molto potente. Tuttavia se un programma funziona, ma non dà risultati corretti, gli errori non potranno essere rivelati con il comando TRACE.
Una volta dato il comando TRACE esso continua a funzionare. Quando avete terminato di tracciare un programma potete disattivare tale comando scrivendo UNTRACE.
Un altro strumento molto utile è il comando BREAK. Facendo riferimento sempre allo stesso programma, se scrivete

```
BREAK 130,150
```

il computer si fermerà non appena incontrerà la linea 130.
A questo punto potete operare nel modo immediato. Allora, se scrivete PRINT X il computer scriverà l'attuale valore di X. Ciò vi permette di conoscere i valori di tutte le variabili del programma e, se necessario, cambiarli. Quando siete pronti per proseguire scrivete CON (abbreviazione di *continue*) e il computer andrà avanti da dove si era fermato. Ogni volta che vengono incontrati dei BREAK e il computer si ferma, questi comandi vengono rimossi. Di conseguenza, se il computer tornerà a ripetere ciclicamente una parte di programma contenente dei breakpoint, questi fermeranno il computer solo la prima volta che verranno incontrati. Ulteriori precisazioni verranno fatte più avanti. Per togliere il BREAK, scrivete UNBREAK. Se volete rimuovere solo certi breakpoint, potete specificare quelli che desiderate. Ad esempio potreste scrivere

```
UNBREAK 140
```

TRACE, UNTRACE, BREAK e UNBREAK possono essere usati anche come istruzioni Basic in un programma. Immaginate di avere dei sospetti sul corretto funzionamento di una parte di un programma.
Usando queste istruzioni nel programma stesso potreste localizzare gli errori. Un esempio di come si potrebbe procedere è

```
(prima parte del programma)
        ...
540 TRACE
        ...
620 BREAK
        ...
780 UNTRACE
        ...
(ultima parte del programma)
```

In questo esempio ipotetico, quando verrà raggiunta la linea 540, verrà attivata la funzione TRACE, pertanto saranno stampati i numeri di linea di tutte le istruzioni che il computer incontra durante l'esecuzione del programma. Quando il computer avrà raggiunto la linea 620, il comando BREAK provocherà uno stop e saremo liberi di indagare sui valori delle variabili usando i comandi immediati. Quando scriviamo CON, il programma proseguirà da dove si era interrotto. Infine, dalla linea 780, verrà disattivata la funzione TRACE.
I comandi BREAK e TRACE forniscono dei mezzi che permettono di seguire l'esecuzione di programmi anche complicati e di vedere esattamente quello che sta accadendo. A questo aggiungete l'altra possibilità data dal metodo con "carta e matita" e voi dovreste essere in grado di correggere ogni programma!
Il tempo che passate ad analizzare nei dettagli i programmi, non è tempo perso. Ne spendereste molto di più a dover correggere gli errori in una fase successiva.

PROBLEMI

1. Scrivete un programma che riceva in input due numeri. Fate stampare il maggiore dei due.

2. Scrivete un programma che legga con READ tre numeri da un'istruzione DATA e stampi il maggiore.

3. Scrivete un programma che calcoli e stampi la somma di tutti i numeri interi da 1 a 100, inclusi.

4. Descrivete con parole vostre cosa succederà se si esegue il seguente programma.

```
100 LET S=0
110 LET X=1
120 LET S=S+X
130 LET X=X+2
140 IF X<100 THEN 120
150 PRINT S
160 END
```

5. Inserite questa istruzione al posto del DATA dell'esempio 3 di questo capitolo.

```
190 DATA 4,2,3,6,5,1111
```

Seguite il programma passo per passo e scrivete cosa succederà in esecuzione.

6. Seguite passo passo il programma seguente usando gli input indicati. In ogni caso individuate cosa verrà stampato. Ecco gli input

a. 1, 2, 3
b. 3, 2, 1
c. 2, 2, 2
d. 3, 1, 3

```
100 INPUT A,B,C
110 IF A<B THEN 150
120 IF B>=C THEN 170
130 LET D=A+B+C
140 GOTO 180
150 LET D=A*B-C
160 GOTO 180
170 LET D=A+B*C
180 PRINT D
190 END
```

7. Supponete di avere un'istruzione DATA contenente una lista di numeri di lunghezza sconosciuta. Tuttavia, la fine della lista è segnalata con il valore convenzionale 9999. Scrivete un programma che calcoli e stampi la somma dei numeri della lista che sono compresi tra −10 e +10, compresi.

8. Una interessante successione di numeri è quella dei numeri di Fibonacci. I primi due sono 0 e 1. Ogni numero successivo della sequenza è la somma dei due precedenti. La serie di Fibonacci è cioè 0, 1, 1, 2, 3, 5, 8,... ecc. Scrivete un programma che calcoli e stampi i primi venti numeri della successione di Fibonacci.

9. Scrivete un programma che accetti due numeri in input. Se entrambi

i numeri sono maggiori o uguali a 10, dovrà stampare la loro somma. Se sono minori di 10, stamperà il prodotto. Se uno è maggiore o uguale a 10 e l'altro è minore di 10, dovrà stampare la differenza tra il maggiore e il minore.

10. Per una interrogazione, un insegnante decide di assegnare dei voti in lettere secondo questo criterio:

90 - 100	A
80 - 89	B
60 - 79	C
50 - 59	D
0 - 49	F

Scrivete un programma che, quando eseguito, fornisca un output di questo tipo:

```
QUAL E' IL VOTO ? 73
CORRISPONDE AL GRADO C
```

11. Se ogni anno il vostro consumo di elettricità cresce dell'8 per cento, in nove anni sarà raddoppiato. Così il vostro "tempo di raddoppiamento" è nove anni. C'è una legge interessante, chiamata "regola del settantadue", che serve per calcolare i tempi di raddoppiamento. Se una certa quantità cresce dell'R per cento in un singolo periodo di tempo, allora il tempo necessario perché la quantità sia raddoppiata sarà circa 72/R. Questa è la regola del settantadue. Possiamo calcolare direttamente l'andamento di un certo processo con il computer. In un singolo periodo di crescita, una quantità aumenta secondo la relazione

$$Q_{nuovo} = Q_{vecchio} (1 + R/100)$$

Così possiamo analizzare la crescita usando più volte questa relazione. Quando il valore di Q è il doppio di quello di partenza, il valore corrispondente dei periodi di crescita sarà il tempo di raddoppiamento. Scrivete un programma che, usando queste idee, dia un output di questo tipo

```
TASSO DI CRESCITA (%) ? 3
IL VALORE DI PERIODI PER
RADDOPPIARE IL VALORE E' 24
```

Usate il programma per verificare l'esattezza della regola del settantadue con diversi tassi di crescita.

12. Degli interi vengono scelti a caso dall'insieme 1, 2, 3 e 4, e posti in un'istruzione DATA. La fine di questi numeri è segnalata dal valore 9999. Scrivete un programma che calcoli e stampi il numero di volte in cui appaiono rispettivamente l'1, il 2, il 3 e il 4. Provate il vostro programma con la seguente istruzione DATA:

```
DATA 3,1,2,1,4,4,1,2,2,2,3,9999
```

ESERCIZI

Controllate i vostri progressi risolvendo i seguenti esercizi. Troverete le risposte alla fine del libro.

1. Cosa darà, in esecuzione questo programma?

```
100 LET Y=3
110 LET X=2*Y
120 PRINT X
130 LET Y=Y+2
140 IF Y<=10 THEN 110
150 END
```

2. Quale sarà l'output se eseguiamo il seguente programma?

```
100 READ X
110 DATA 1,2,3
120 IF X<2 THEN 160
130 IF X=2 THEN 150
140 PRINT "BUONO"
150 PRINT "DISTINTO"
160 PRINT "OTTIMO"
170 PRINT
180 GOTO 100
190 END
```

3. Supponiamo che vogliate comperare un certo numero di articoli. Il produttore fissa i prezzi in base alla quantità acquistata. Ecco il dettaglio dei prezzi:

Quantità di articoli	Prezzo di un articolo
20 o meno	2000
da 20 a 50	1800
51 o più	1500

Scrivete un programma che, quando eseguito, fornisca il seguente tipico output:

```
QUANTI SONO GLI ARTICOLI? 40
IL PREZZO DI UN ARTICOLO E' 1800
LA SPESA TOTALE E' 72000
```

Costruite un ciclo che permetta di ripetere il procedimento.

4. Scrivete un programma che stampi la seguente tabella di numeri e poi si fermi. I numeri siano disposti secondo la spaziatura standard.

```
0                   5
10                  15
20                  25
          ECC.
100                 105
```

5. Se ricevete una multa per eccesso di velocità, supponiamo che la multa dipenda da quanto avete superato i limiti. Ecco come calcolarla:

Eccesso di velocità	Multa
1 – 10 Km/h	5000
11 – 20	10000
21 – 30	20000
31 – 40	40000
41 o più	80000

Scrivete un programma che, quando eseguito, fornisca il seguente tipico output:

```
LIMITE DI VELOCITA' ? 45
VELOCITA' EFFETTIVA ? 56
LA MULTA E' DI 10000 LIRE
```

CAPITOLO SETTE

Cicli e funzioni

ARGOMENTI

Vedremo in questo capitolo due interessanti caratteristiche del Basic, che ci forniranno nuove potenti possibilità di programmazione. Ecco i nostri argomenti:

I cicli FOR...NEXT

Abbiamo già imparato come si costruiscono cicli in un programma usando le istruzioni di salto incondizionato e condizionato. Il Basic possiede delle istruzioni speciali che formano automaticamente un ciclo. Esse consentono di programmare più semplicemente e con maggiori possibilità.

Le funzioni interne

Nel Basic ci sono anche delle funzioni interne che possono essere richiamate per scopi specifici. Esamineremo alcune tra le più semplici di esse e vedremo come devono essere usate nei programmi in Basic.

Applicazioni

Continueremo con degli esercizi per introdurvi nel mondo della programmazione. Ricordiamo infatti che l'obiettivo principale del libro è quello di insegnare a scrivere programmi in linguaggio Basic per il TI 99/A home computer.

PRATICA SUL CALCOLATORE

Cominciamo subito a lavorare sul calcolatore.

1. Accendete il vostro computer, selezionate il Basic e scrivete questo programma:

```
100 LET Y=10
110 PRINT Y,
120 LET Y=Y+5
130 IF Y<=50 THEN 110
140 END
```

Esaminate il programma e fatelo funzionare. Prendete nota di quello che succede.

____

Quale istruzione del programma fa sì che i numeri stampati siano diversi tra loro?

____

2. Cancellate il programma e liberate il video. Scrivete ora questo altro programma:

```
100 FOR Y=10 TO 50 STEP 5
110 PRINT Y,
120 NEXT Y
130 END
```

Fatelo eseguire e riportate il risultato.

____

Confrontate l'output con quello del programma al passo 1.

3. Poiché i due programmi che avete eseguito danno lo stesso risultato, si è portati a pensare che le loro istruzioni siano in qualche modo

correlate. Modificate la linea 100 in questo modo

```
100 FOR Y=10 TO 50 STEP 10
```

Fate una lista del programma ed esaminatelo. Quale risultato vi aspettate dall'esecuzione del programma?

Controllate l'esattezza di ciò che avete previsto. Eseguite il programma e segnate il risultato.

---

4. Cerchiamo di modificare un po' il programma. Cambiamo la riga 100 così

```
100 FOR Y=0 TO 5 STEP 1
```

Listate il programma. Cosa pensate che farà?

Eseguite il programma e trascrivete l'output.

---

5. Riscrivete la riga 100 così

```
100 FOR Y=0 TO 5
```

Fate ancora una lista. Cosa farà quest'altro programma?

Eseguitelo e annotate cosa è successo.

Confrontate la riga 100 del programma appena eseguito con quella del programma al passo 4. È necessaria l'istruzione STEP quando l'incremento tra i numeri da stampare è 1?

6. Usiamo un'altra tattica. Cambiamo la riga 100 in questo modo

```
100 FOR Y=20 TO 10 STEP -2
```

Fate una lista del programma ed esaminatelo. Quale risultato darà?

Eseguite il programma e trascrivete l'output.

---

7. Bene, scrivete ora così la riga 100

```
100 FOR Y=10 TO 20 STEP -2
```

Fate una lista. Cosa succederà se eseguite il programma?

Fatelo funzionare e scrivete cosa è accaduto.

Vi abbiamo mostrato con ciò un errore che è possibile commettere in Basic. Qual è il problema?

---

8. Finora gli incrementi delle istruzioni FOR...NEXT non creavano problemi. Proviamo adesso con un incremento che non incontri esattamente i limiti dell'istruzione FOR...NEXT. Cambiamo la riga 100 così

```
100 FOR Y=2 TO 9 STEP 3
```

Listate il programma. Scrivete quello che secondo voi apparirà.

Eseguite il programma e riportate quello che è successo.

---

9. Continueremo ora con dei casi un po' più complicati, sempre riguardanti le istruzioni FOR...NEXT. Cancellate il programma dalla memoria con il comando NEW e scrivete il seguente programma:

```
100 FOR X=1 TO 3
110 FOR Y=1 TO 4
120 PRINT X,Y
```

```
130 NEXT Y
140 NEXT X
150 END
```

Eseguitelo e scrivete il risultato.

---

10. Cambiate la riga 100 con questa

```
100 FOR X=1 TO 2
```

Fate funzionare questo nuovo programma e trascrivete l'output.

Confrontate i risultati dei due esempi. Riuscite a vedere la connessione tra le liste e le limitazioni delle istruzioni FOR...NEXT?

---

11. Modifichiamo ulteriormente il programma. Cambiate le righe 100 e 110 con queste

```
100 FOR X=1 TO 3
110 FOR Y=1 TO 2
```

Listate questo programma ed esaminatelo. Cosa pensate che stamperà se lo eseguite?

Fatelo e controllate se avete sbagliato.

---

12. Cambiamo ancora. Scrivete così le righe 100 e 110

```
100 FOR X=1 TO 2
110 FOR Y=1 TO 2
```

Fate una lista del programma e segnatevi cosa succederà, secondo voi, se lo fate funzionare.

Eseguite il programma e riportate sotto il risultato.

Azzerate il video e listate il programma. Tracciate idealmente una graffa dall'istruzione FOR X al NEXT X. Fate la stessa cosa da FOR Y a NEXT Y. Si incrociano queste due parentesi?

---

13. Cambiamo ora in questo modo le righe 100 e 110

```
100 FOR Y=1 TO 2
110 FOR X=1 TO 2
```

Fate una lista del programma. Quale sarà l'output del programma?

Fatelo funzionare e annotate cosa è successo.

Azzerrate il video e listate il programma. Di nuovo tracciate una linea immaginaria tra le istruzioni FOR X e NEXT X. Fate lo stesso per FOR Y e NEXT Y come al punto 12. Si intersecano le due righe? Confrontate la situazione con quella del passo 12.

Tutto questo vi suggerisce un modo per evitare di far confusione usando più di un ciclo FOR...NEXT in un solo programma?

14. Nel capitolo 5 abbiamo imparato ad usare la funzione TAB per incolonnare un output a nostro piacimento. Ora che disponiamo delle istruzioni FOR...NEXT riconsideriamo per un attimo la funzione TAB. Cancellate il programma dalla memoria, liberate il video e scrivete il seguente programma:

```
100 FOR X=1 TO 5
110 PRINT TAB(X);
120 FOR Y=X TO 5
130 PRINT "Y";
140 NEXT Y
150 PRINT
160 NEXT X
170 END
```

Controllate un attimo il programma seguendo la tecnica sviluppata nell'ultimo capitolo. Esaminatelo passo per passo scrivendo i valori che le variabili assumono via via. Quale risultato otterrà questo programma?

___

Controllate se avete detto giusto. Eseguite il programma e prendete nota del risultato.

___

15. Cancellate il programma che c'è in memoria. Liberate il video e scrivete quest'altro programma:

```
100 INPUT A
110 B=SQR(A)
120 PRINT B
130 GOTO 100
140 END
```

Eseguitelo e, quando appare il segnale di input, scrivete 4. Cosa è successo?

___

Ora scrivete 9 e segnate il risultato.

___

Ancora, dategli 25. Cosa risulta?

___

Ed ora 10, per l'ultima volta. Cosa è accaduto?

___

Cioè cosa succede ad A nell'espressione SQR(A) della riga 110 del programma? In altre parole, cosa fa SQR?

___

16. Fate uscire il computer dal ciclo di input. Cambiate ora la riga 110 con

```
110 LET B=INT(A)
```

Eseguite il programma per i seguenti valori di A. Per ognuno di essi segnate il risultato.

A	Output
1	________
3.4	________
256.78	________
0	________
−1	________
−2.3	________

Esaminate i risultati che avete trovato e paragonateli al corrispondente valore di A. Cosa fa la funzione INT(A)?

____

Se avete avuto dei problemi nel capire cosa è accaduto ai valori di A negativi, non è il caso di preoccuparvi, per adesso. Più tardi rivedremo tutto.

17. Fate uscire il computer dal ciclo di input. Sostituite la riga 110 con quella che segue

```
110 LET B=SGN(A)
```

Listate il programma. Riguardate la struttura del programma per ricordare bene come funziona. Eseguitelo per ognuno dei valori seguenti di A. In ogni caso segnate il risultato.

A	Output
1.5	________
43	________
128.3	________
0	________
−1	________
−1.2	________
−345.7	________
4.7	________
−5.8	________

Esaminate attentamente l'output. Cosa fa la funzione SGN?

18. Consideriamo un'altra funzione. Fate uscire il calcolatore dal ciclo di input. Cambiate la riga 110 in

```
110 LET B=ABS(A)
```

Eseguite il programma per tutti i valori di A elencati sotto. Come prima, scrivete tutti i risultati.

A	Output
3.4	________
0	________
−3.4	________
−2	________
2	________
−8.45	________
8.45	________

Analizzate i risultati. Cosa fa la funzione ABS?

---

19. Torniamo al concetto di variabile stringa, che abbiamo precedentemente introdotto. Più precisamente, vogliamo analizzare le caratteristiche di alcune funzioni che lavorano sulle stringhe di caratteri. Cancellate il programma dalla memoria, liberate il video e scrivete questo programma:

```
100 LET A$="CALCOLATORE"
110 LET B$="ELETTRONICO"
120 LET C$=SEG$(A$,1,2)
130 PRINT C$
140 END
```

Il nostro argomento è la funzione SEG$ della riga 120. Sapete indovinare qual è il suo scopo?

Eseguite il programma e scrivete ciò che il computer ha stampato.

---

20. Bene, ora modificate l'istruzione SEG$ nella riga 120 così

```
120 LET C$=SEG$(A$,1,4)
```

Eseguite il programma e annotate cosa è successo.

Avete già capito quello che fa la funzione SEG$?

---

21. Proviamo ancora una volta. Cambiamo la funzione SEG$ della riga 120 con

```
120 LET C$=SEG$(B$,3,4)
```

Cosa accadrà ora?

Fate eseguire il programma e controllate se avete indovinato.

22. Cambiamo la riga 120 come segue:

```
120 LET C$=SEG$(A$,10,1)
```

Quale sarà l'output?

Controllate se la vostra risposta è esatta. Prendete nota del risultato.

---

23. Modificate la funzione SEG$ così

```
120 LET C$=SEG$(B$,2,5)
```

Cosa stamperà ora il computer?

Eseguite il programma e scrivete quello che è successo.

Fino ad ora ci siamo fatti un'idea soddisfacente dello scopo dell'istruzione SEG$. Ora proviamo in qualche altro modo. Cambiate la riga nel modo seguente:

```
120 LET C$=SEG$(A$,8,10)
```

Abbiamo chiesto una parte della stringa maggiore della stringa stessa. Cosa pensare che accadrà?

Eseguite il programma e riportate il risultato.

---

25. Infine, modificate la funzione SEG$ nella riga 120 con

```
120 LET C$=SEG$(B$,5,-2)
```

Cosa succederà ora?

Eseguite il programma e segnate il risultato.

---

26. Questo può bastare per quanto riguarda la funzione SEG$. Cancellate il programma in memoria ed inserite il seguente:

```
100 INPUT A$
110 PRINT LEN(A$)
120 GOTO 100
130 END
```

Eseguite il programma e quando apparirà il segnale di input, scrivete AUTOMOBILE. Cosa è successo?

---

27. Scrivete una parola di lunghezza differente. Cosa appare sul video?

Provate con un certo numero di parole differenti. Qual è lo scopo dell'istruzione LEN?

---

28. Uscite dal ciclo di input. Cancellate il programma dalla memoria. Ora inserite quello che segue.

```
100 INPUT N
110 PRINT CHR$(N)
120 GOTO 100
130 END
```

La nuova istruzione in questo programma è CHR$. Fate funzionare il programma e, al segnale di input, scrivete il numero 65. Che cosa è successo?

---

29. Il computer è in attesa di un altro numero. Questa volta scrivete 91. Che cosa appare?

Avete capito lo scopo della funzione CHR$?

---

30. Sperimentate questo programma. Inserite numeri da 33 a 91. Dovreste accorgervi abbastanza rapidamente di cosa sta accadendo. Spiegate con parole vostre (se potete) la funzione del comando CHR$.

---

31. Uscite dal ciclo di input. Cancellate il programma dalla memoria e scrivete il seguente:

```
100 INPUT A$
110 LET N=VAL(A$)
120 PRINT N
130 GOTO 100
140 END
```

In questo programma esamineremo l'istruzione VAL presente nella linea 110. Notate che il programma chiede l'inserimento di una stringa. La stringa A$ viene manipolata dalla funzione che ricava un risultato numerico da assegnare alla variabile numerica N, valore che verrà poi stampato. Fate funzionare il programma e al segnale di input, scrivete il numero 25 (ricordate che il computer lo sta trattando come una stringa). Cosa è accaduto?

---

32. Provate ad inserire numeri differenti. Annotate ogni volta il risultato.

____

Vi siete già fatti un'idea dello scopo della funzione VAL?

____

33. Facciamo un tentativo differente. Scrivete **A3B6**. Cosa è successo?

____

Per ora può bastare. Speriamo che abbiate cominciato a capire come funziona l'istruzione VAL. La discuteremo in modo completo più avanti nello stesso capitolo. Uscite dal ciclo di input.

34. Consideriamo ora un'altra istruzione. Cancellate il programma in memoria e scrivete il seguente.

```
100 INPUT N
110 LET A$=STR$(N)
120 PRINT A$
130 GOTO 100
140 END
```

Questo programma fa l'inverso del precedente, visto al punto 31. Quel programma chiedeva una stringa e stampava un numero. Questo chiede un numero e stampa una stringa. Eseguite il programma e quando vedrete il segnale di input, scrivete **45**. Cosa è apparso sul video?

____

35. Provate con numeri differenti. Cosa viene stampato di volta in volta?

____

Immaginate già lo scopo di STR$?

____

36. Questa volta scrivete ABC. Cosa è successo?

____

Torneremo sullo stesso argomento più avanti. Ciò che appare chiaramente è che VAL e STR$ sono due istruzioni in stretta relazione. Fate uscire il computer dal ciclo di input.

37. Cancellate il programma dalla memoria ed inserite il seguente:

```
100 LET A$="MISSISSIPPI"
110 LET B$="IS"
120 PRINT POS(A$,B$,1)
130 END
```

Eseguite il programma e annotate qui sotto il risultato.

---

38. Cambiate l'istruzione POS di linea 120 in questo modo: POS (A$,B$,3). Ora eseguite il programma e scrivete che cosa è successo.

---

39. Infine, cambiate B$ di linea 110 in SI ed eseguite il programma. Qual è il risultato?

Capite lo scopo dell'istruzione POS? In caso negativo non preoccupatevi. Riprenderemo l'argomento più avanti.

---

40. Questo conclude la pratica al computer per ora. Spegnete il calcolatore e passate alla discussione.

DISCUSSIONE

Le tecniche che abbiamo utilizzato nel lavoro sul calcolatore possono rendere i nostri programmi più potenti. Per poter trarre il massimo vantaggio dal loro uso dobbiamo però capire esattamente come funzionano.

La costruzione di cicli FOR...NEXT

Nei capitoli precedenti abbiamo studiato come si realizzano i cicli per mezzo delle istruzioni di salto. Abbiamo visto che il salto incondizionato

(GOTO), pur essendo molto utile, dava luogo di solito ad un ciclo senza via d'uscita. Il salto condizionato (IF...THEN) invece, ci permetteva anche di uscire dal ciclo al momento opportuno. Ambedue queste tecniche sono buone. Tuttavia, il Basic può provvedere alla costruzione dei cicli in un modo molto elegante, che evita fatica al programmatore. Vediamo in che cosa consiste questo metodo, che fa uso delle istruzioni FOR NEXT. Tutte le istruzioni FOR hanno lo stesso formato, che è mostrato di seguito assieme ad un esempio di istruzione.

Numero di linea FOR <variabile> = <relazione> TO <relazione> STEP <relazione>

```
120 FOR X=1 TO 9 STEP 2
```

Le sole cose che possono cambiare o assumere un'altra forma nelle istruzioni FOR sono la variabile e le tre espressioni. Se l'istruzione STEP viene tralasciata, l'incremento usato dal computer sarà 1. Possiamo scrivere l'istruzione FOR in modi diversi. Alcuni di essi sono elencati sotto per suggerire alcune possibilità di utilizzo di questa istruzione.

```
130 FOR J=2 TO 8
130 FOR T=25 TO 10 STEP -2
130 FOR W=-20 TO 10 STEP 2
130 FOR X=3*Z TO A*B STEP D
```

In generale una qualsiasi istruzione corretta del Basic può essere assunta come espressione da inserire nell'istruzione FOR, purché le variabili usate siano state opportunamente definite nel programma.
L'istruzione FOR serve per aprire un ciclo che verrà chiuso dall'istruzione NEXT. Il primo esempio mostra come si realizza tutto ciò.

```
200 FOR X=2 TO 18 STEP 2   (apre il ciclo)
                  *
                  *
                  *
     Righe di programma contenute nel ciclo
                  *
                  *
                  *
340 NEXT X   (chiude il ciclo)
```

La variabile che compare dopo NEXT deve essere la stessa usata nell'istruzione FOR che apriva il ciclo.
È importante capire nei dettagli come funzionano questi cicli. Nell'esempio sopra, quando il calcolatore raggiunge la riga 200 per la prima volta, pone X uguale a 2. Poi esegue le righe che precedono la 340. Questa chiude il ciclo e rispedisce il computer alla riga 200, dove X assume il prossimo valore, che in questo caso è 4. Il calcolatore percorre il ciclo finché X non raggiunge o supera il limite 18. Allora, invece di ripetere tutte le operazioni del ciclo, il calcolatore passa alla riga che segue

l'istruzione NEXT, che chiude il ciclo. Ecco un altro esempio che ci mostra in azione le istruzioni FOR...NEXT:

```
100 LET A=1
110 FOR X=1 TO 6 STEP 2
120 LET A=2*A
130 PRINT A,X
140 NEXT X
150 END
```

Giacché questo programma contiene solo due variabili (A ed X), possiamo fare una lista delle etichette, nell'ordine in cui il computer le esamina, e dei corrispondenti valori delle variabili.

Etichette	A	X
100	1	
110	1	1
120	2	1
130	2	1
140	2	3
120	4	3
130	4	3
140	4	5
120	8	5
130	8	5
140	8	7 (Si esce dal ciclo)
150	(Il programma si ferma)	

Analizzate la sequenza delle etichette e dei relativi valori delle variabili A ed X finché avete capito bene in che modo le istruzioni FOR...NEXT controllano il ciclo. Piuttosto spesso in un programma si richiedono combinazioni di cicli più complicate. La struttura può essere complessa quanto si vuole, purché i cicli non si intersechino. Ecco un esempio di come due cicli si intersecano:

```
100 FOR I=0 TO 20 STEP 2
110 FOR A=10 TO 2 STEP -1
120 FOR B=1 TO 4
170 NEXT A
180 NEXT B
190 NEXT I
```

Il ciclo esterno va bene, quelli interni si incrociano!

Un altro esempio di cicli che si incrociano è questo

Ecco ora un esempio di struttura complessa in cui i cicli sono organizzati correttamente:

In questo esempio ci sono cicli doppi e nidi di cicli, cioè cicli uno interno all'altro. Si ricordi sempre che in un programma si può usare una qualsiasi combinazione, purché le linee che collegano le istruzioni FOR con i corrispondenti NEXT non si intersechino. Se lo fanno, il computer segnalerà un errore e si fermerà.

Le funzioni interne

Uno dei vantaggi di un moderno computer digitale è quello di poter eseguire certi compiti per mezzo di insiemi di istruzioni programmate a parte. Siccome ci sono molti calcoli che si devono spesso ripetere, i costruttori li hanno preparati svolti sotto forma di funzioni. Servendosi di esse, il programmatore in Basic può svolgere operazioni molto complesse senza difficoltà. Vedremo parecchie di queste funzioni ed il loro uso corretto.

Funzione	Operazione
SQR(X)	Radice quadrata di X
INT(X)	Parte intera di X
SGN(X)	Segno di X
ABS(X)	Valore assoluto di X

Usiamo la prima funzione, SQR(X), per capire come le funzioni, in genere, si comportano. Come prima cosa, X è detto l'"argomento" della funzione. Se questa definizione non vi dice niente pensate ad X come a "ciò su cui agisce la funzione". Quando, in un programma, usiamo SQR(X), diciamo al computer di prendere il valore di X e di farne la radice quadrata. Per esempio

```
SQR(36)  = 6
SQR(64)  = 8
SQR(100) = 10
SQR(2)   = 1.41421
```

e così via. L'unica limitazione è che non si può fare la radice quadrata di un numero negativo. Se il calcolatore dovesse fare, per esempio, SRQ(−6) segnalerebbe un errore e quindi si fermerebbe.
Come argomento può essere presa qualsiasi espressione che il programma richiede. Quando il computer incontra un'espressione come SQR(X+4*Y), prende i valori di X e Y, fa i calcoli racchiusi in parentesi e fa la radice quadrata del risultato. Questo vale per tutte le funzioni.
INT(X) dà la parte intera di X. INT è l'abbreviazione di intero. Ad esempio, 2 è un intero, mentre 23.475 non lo è. Se cerchiamo la parte intera di un numero positivo, semplicemente tralasciamo ciò che segue il punto decimale. Così

```
INT(3.1593)  = 3
INT(54.76)   = 54
INT(0.362)   = 0
```

I numeri negativi, invece, richiedono un'attenzione speciale. Ciò che in effetti si fa quando si calcola la parte intera di un numero è prendere il più grande intero minore di quel numero. In questo modo si capisce che

INT(−2) = −2
INT(−.93) = −1

e così via. Si noti bene che la funzione INT non arrotonda il numero. Spesso chi la vede per la prima volta confonde le due cose.
SGN(X) è una funzione molto interessante. Se X (l'argomento della funzione) è positivo, SGN(X) dà +1. Se X è negativo SGN(X) dà −1. Se SGN(X) è zero, SGN(X) dà 0. Cioè SGN(X) dà effettivamente il segno di X, +1, −1 o 0. Quindi

SGN(4.568) = +1
SGN(375) = +1
SGN(0) = 0
SGN(−5.9031) = −1
SGN(−4) = −1

A questo punto potrebbe non esservi chiara l'utilità di questa funzione. Sappiate, comunque, che è molto utile in diverse occasioni. Per ora accontentiamoci di sapere come funziona.
ABS(X) dice semplicemente al computer di ignorare il segno di X. Cioè di tradurre tutti i valori di X in numeri positivi. Così

ABS(4.5) = 4.5
ABS(−4.5) = 4.5
ABS(95.34) = 95.34
ABS(−95.34) = 95.34
ABS(0) = 0

Le funzioni che agiscono su stringhe di caratteri sono interessanti e molto utili. La prima di esse, SEG$(A$,M,N) dice al computer di prendere N caratteri dalla stringa A$ cominciando dall'Mesimo carattere. SEG sta per "segmento" e naturalmente si riferisce a un segmento di una stringa di caratteri. L'istruzione SEG$ può essere usata con ogni stringa.
Per vedere come funziona, poniamo B$ = "TELEVISIONE", allora

```
SEG$(B$,1,3)="TEL"
SEG$(B$,5,1)="V"
SEG$(B$,3,8)="LEVISION"
```

Nell'esempio abbiamo messo le virgolette, ma esse non fanno parte delle

sottostringhe. Come potete immaginare la possibilità di operare con segmenti di stringa apre la strada a nuove applicazioni.
Un'altra funzione incontrata nelle attività pratiche è l'istruzione LEN. È molto semplice da spiegare poiché essa dà il numero di caratteri che formano una stringa. Ad esempio, se T$ = "ORITTEROPO", allora LEN(T$) = 10. Le virgolette che racchiudono la stringa non vengono contate come caratteri.
Spesso, quando stiamo lavorando con stringhe di lunghezza sconosciuta o variabile, l'istruzione LEN è veramente indispensabile!
L'istruzione CHR$ viene usata per generare caratteri di una lista codificata dall'industria elettronica. Questa lista (detta insieme di caratteri ASCII) comprende 127 caratteri che sono contraddistinti da un numero. Ad esempio

```
100 LET A$=CHR$(N)
```

assegnerà l'ennesimo carattere dell'insieme ASCII alla variabile A$. La lista completa dei caratteri ASCII è riportata nel manuale del vostro home computer.
Durante le attività pratiche sul computer probabilmente avevate imparato a grandi linee come usare i caratteri ASCII. CHR$(65) è A e CHR$(90) è Z. Tutte le lettere maiuscole sono comprese tra questi due numeri. Il numero 0 è il carattere con codice 48; 9 è il carattere numero 57. È preferibile che vi riferiate al vostro manuale per avere la lista completa dei caratteri. Per il momento accontentiamoci di considerare lo scopo della funzione CHR$ e la sua connessione con il set di caratteri ASCII.
Ci sembra opportuno discutere una istruzione non considerata in precedenza e che è in stretta relazione con la funzione CHR$. È questa la funzione ASC. Una tipica istruzione potrebbe essere

```
200 LET X=ASC(A$)
```

Questa fa sì che il codice ASCII del primo carattere di A$ venga assegnato ad X. Se A$ corrisponde ad HOTEL, alla X verrà assegnato il valore 72 che è il numero di codice ASCII di H.
Le prossime due funzioni che prenderemo in considerazione sono VAL e STR$. Prima di discuterle dobbiamo ricordare un fatto importante. Il numero 25 può essere sia il numero venticinque sia la stringa di caratteri 25. Il computer tratta in modo differente i numeri e le stringhe. Ad esempio, il numero 25 richiede maggior spazio nella memoria rispetto ai caratteri 2 e 5. Per questa ragione se la memoria è limitata potete risparmiare spazio collocando i numeri in forma di stringa.
I numeri possono essere cambiati in stringhe con la funzione STR$. Ad esempio

```
120 LET A$=STR$(17)
```

converte il numero 17 nella stringa di caratteri 17. Spesso si può fare un po' di confusione poiché la rappresentazione sullo schermo dei numeri in forma numerica o di stringa è esattamente la stessa. Invece per il computer sono due quantità completamente differenti.
La funzione VAL converte un numero dalla rappresentazione in forma di stringa a quella numerica. Perciò la parte di programma

```
150 LET A$="23"
160 N=VAL(A$)
```

converte la stringa 23 nel numero 23 ed assegna questo valore alla variabile N. Naturalmente se tentiamo di cambiare 28B in una rappresentazione numerica, il computer oltre alle cifre troverà un carattere (la lettera B), segnalerà un errore e si fermerà.
L'ultima funzione che discuteremo è l'istruzione POS.
La sua forma è sempre POS(A$,B$,X). In un primo tempo sembra complicata ma non è difficile da capire. Se N = POS(A$,B$,X), N sarà il numero della posizione in cui si trova la stringa B$ nella stringa A$, la ricerca verrà iniziata dalla posizione indicata dalla X. Allora se A$ = "AUTOMOBILE" e B$ = "O", POS(A$,B$,1) = 4 e POS(A$,B$,5) = 6. Se la ricerca non ha esito, la funzione ritorna il valore zero. Nell'esempio precedente, POS(A$,B$,7) = 0.
Nel TI Basic esistono altre funzioni. Tuttavia la maggior parte di esse implica conoscenze matematiche che vanno oltre lo scopo di questo libro. Se possedete le necessarie cognizioni matematiche non avrete alcuna difficoltà a capire il loro significato ed il loro uso. Se siete interessati consultate il manuale del vostro computer.
Alcune funzioni che abbiamo già considerato vengono usate come istruzioni Basic. Riportiamo alcuni esempi di linee di programma che utilizzano tali funzioni

```
100 LET X=SQR(Y)
100 LET Z=3*INT(C)+ABS(D)
```

Tali funzioni possono essere usate all'interno di altre istruzioni. Un esempio è

```
100 LET Y=INT(SQR(X)+3*ABS(Z))
```

ESEMPI DI PROGRAMMI

Gli esempi che vedremo sono stati scelti per mostrare come l'uso dei cicli FOR...NEXT e delle funzioni interne può semplificare la vita al programmatore.

Esempio 1 - Calcolo della media

Abbiamo già visto come esempio il programma del calcolo di una media nel capitolo precedente. Consideriamo ora lo stesso problema, ma usiamo un metodo differente. Vogliamo che il programma stampi, in esecuzione, qualcosa di questo tipo:

```
QUANTI SONO I NUMERI ? 5
INTRODUCI I NUMERI, UNO ALLA
 VOLTA
? 12.5
? 10.8
? 11.3
? 14.1
? 12.8
LA MEDIA E' 12.3
```

Adesso dovrebbe essere facile per voi scrivere le prime righe.

```
100 PRINT "QUANTI SONO I NUM
ERI";
110 INPUT N
120 PRINT "INTRODUCI I NUMER
I,"
130 PRINT "UNO ALLA VOLTA"
```

Dobbiamo trovare il modo di introdurre N numeri, ma bisogna tener presente che ciò che vogliamo è poi la media di questi numeri. Così poniamo inizialmente S (che sarà usato come somma dei numeri) uguale a 0.

```
140 LET S=0
```

L'input degli N numeri e la somma possono essere svolti ottimamente con le istruzioni FOR...NEXT.

```
150 FOR I=1 TO N
160 INPUT X
170 LET S=S+X
180 NEXT I
```

Si noti che usiamo I, la variabile del ciclo, solo per contare i numeri che

sono stati introdotti. Quando ci sono tutti, il computer uscirà dal ciclo proseguendo con la riga che segue quella con etichetta 180. A tal punto S conterrà la somma di tutti i valori che sono stati introdotti. Poiché sappiamo che sono N, possiamo subito calcolare la media.

```
190 LET A=S/N
```

Il resto del programma non contiene difficoltà.

```
200 PRINT "LA MEDIA E'";A
210 END
```

Ecco il programma completo:

```
100 PRINT "QUANTI SONO I NUM
ERI";
110 INPUT N
120 PRINT "INTRODUCI I NUMER
I,"
130 PRINT "UNO ALLA VOLTA"
140 LET S=0
150 FOR I=1 TO N
160 INPUT X
170 LET S=S+X
180 NEXT I
190 LET A=S/N
200 PRINT "LA MEDIA E'";A
210 END
```

Esempio 2 - Tavola di conversione delle temperature

In uno dei programmi precedenti abbiamo usato la relazione

```
C=5/9(F-32)
```

per tradurre i gradi Fahrenheit in gradi Celsius. Vogliamo generare un output come questo:

Gradi F		Gradi C
0		−17.7777
5		−15
10		−12.2222
	ecc.	
100		37.7777

Facciamo stampare prima l'intestazione e gli spazi che precedono la tabella vera e propria.

```
100 PRINT "GRADI F","GRADI C
"
110 PRINT
```

Per generare i valori di F, che poi devono essere tradotti in C, possiamo usare un ciclo FOR...NEXT.

```
120 FOR F=0 TO 100 STEP 5
130 LET C=5*(F-32)/9
140 PRINT F,C
150 NEXT F
```

Infine, ci vuole l'istruzione END.

```
160 END
```

Ecco tutto il programma:

```
100 PRINT "GRADI F","GRADI C
"
110 PRINT
120 FOR F=0 TO 100 STEP 5
130 C=5*(F-32)/9
140 PRINT F,C
150 NEXT F
160 END
```

Esempio 3 - Un problema che riguarda l'alfabeto

Supponiamo di dover scrivere un programma che stampi le seguenti sequenze di lettere:

```
ABCDEF
 BCDEFG
  CDEFGH
   ecc.
```

Il numero delle sequenze deve essere sufficiente per coprire l'intero alfabeto. Per far questo useremo una funzione che lavora su stringhe. Per prima cosa, comunque, definiamo l'alfabeto come un'unica stringa.

```
100 LET A$="ABCDEFGHIJKLMNOP
QRSTUVWXYZ"
```

Analizzando attentamente le sequenze, vedrete che il loro numero deve essere 21. Ogni riga deve contenere sei caratteri. Dobbiamo riuscire a stampare ogni sequenza spostata di uno spazio a destra della precedente. Ecco i comandi necessari per far questo

```
110 FOR I=1 TO 21
120 PRINT TAB(I);
130 PRINT SEG$(A$,I,6)
140 NEXT I
```

La stampa è incolonnata dalla funzione TAB della riga 120. La funzione SEG$ della riga 130 prende i sei caratteri dall'alfabeto.
Con l'aggiunta dell'istruzione END, il programma completo risulta

```
100 LET A$="ABCDEFGHIJKLMNOP
QRSTUVWXYZ"
110 FOR I=1 TO 21
120 PRINT TAB(I);
130 PRINT SEG$(A$,I,6)
140 NEXT I
150 END
```

Questo è un bel programma da provare. Dopo averlo eseguito per vedere che le sequenze di lettere siano giuste, provate a cambiare alcuni parametri nel programma e vedete cosa succede.

Esempio 4 - Piano di deprezzamento

Quando una società investe dei soldi in attrezzature, l'investimento viene svalutato per un certo numero di anni a scopi fiscali. Ciò significa che il valore delle attrezzature è ridotto ogni anno (per il consumo e il deterioramento) e questo decremento è un elemento detraibile dalle tasse. Uno dei metodi usati per calcolare la svalutazione è basato sulla "somma delle cifre degli anni". Per chiarire meglio, supponiamo che un articolo dell'attrezzatura abbia una durata di cinque anni. La somma delle cifre degli anni sarà

```
1+2+3+4+5=15
```

La svalutazione al primo anno sarà 5/15 del valore iniziale; la frazione di svalutazione al secondo anno sarà 4/15; e così via. Ogni anno il valore del bene decresce a causa del suo deprezzamento. Alla fine dell'ultimo anno di vita il suo valore sarà zero.
Vogliamo scrivere un programma Basic per costruire un piano di deprezzamento. Come prima cosa dobbiamo sapere qual è il valore dell'attrezzatura e la sua durata.

```
100 PRINT "VALORE DEL BENE (
LIRE)";
110 INPUT P
120 PRINT "DURATA DEL BENE (
ANNI)";
130 INPUT N
140 PRINT
150 PRINT "ANNI","DEPREZZ.",
"VALORE"
160 PRINT
```

La somma delle cifre degli anni si calcola facilmente.

```
150 LET S=0
160 FOR I=1 TO N
170 LET S=S+I
180 NEXT I
```

Ed ora prepariamo il piano e lo stampiamo. Useremo la variabile P1 per tener conto del valore corrente del bene.

```
190 LET P1=P
200 FOR I=1 TO N
210 LET F=(N+1-I)/S
220 LET D=P*F
230 LET P1=P1-D
270 PRINT "ANNO ";I
280 PRINT "DEPREZZ. ";D
290 PRINT "VALORE ";P1
300 NEXT I
```

Nella riga **210**, **F** è la frazione di svalutazione per l'i-esimo anno. Potete controllarla per i valori di **I** se volete assicurarvi che l'espressione generi il valore corretto di **F**. Nella linea **250 D** è la svalutazione. L'unica cosa che manca è l'istruzione **END**. L'intero programma è

```
100 PRINT "VALORE DEL BENE (
LIRE)";
110 INPUT P
120 PRINT "DURATA DEL BENE (
ANNI)";
130 INPUT N
140 PRINT
150 LET S=0
160 FOR I=1 TO N
170 LET S=S+I
180 NEXT I
190 LET P1=P
200 FOR I=1 TO N
210 LET F=(N+1-I)/S
220 LET D=P*F
230 LET P1=P1-D
270 PRINT "ANNO ";I
280 PRINT "DEPREZZ. ";D
290 PRINT "VALORE ";P1
300 NEXT I
310 END
```

Provate il programma con diversi dati iniziali.

PROBLEMI

1. Scrivete un programma per costruire una tabella di numeri con le relative radici quadrate. Il risultato dovrà essere del tipo:

N	SQR(N)
2.0	1.41421
2.1	1.44914
2.2	1.48324
ecc.	
3.9	1.97484
4.0	2.00000

2. Calcolate l'espressione

$$X^2+3X-4$$

per X = 0, 0.1, 0.2, ..., 1.9, 2.0. Fate stampare i valori di X e i valori corrispondenti dell'espressione sulla stessa riga.

3. Scrivete un programma che accetti un numero N in input e quindi stampi tutti i numeri pari maggiori di 0 e minori o uguali ad N.

4. Scrivete un programma che usi le istruzioni FOR NEXT ... per leggere dieci coppie di numeri con le istruzioni DATA. Per ogni coppia, fate stampare i numeri e la loro somma.

5. Esaminate il seguente programma. Se lo fate funzionare, quale sarà il suo output?

```
100 FOR I=1 TO 5
110 READ A
120 LET B=INT(A)-SGN(A)*2
130 PRINT B
140 NEXT I
150 DATA 2.2,-3,10,0,-1.5
160 END
```

6. Spiegate qual è lo scopo di questo programma:

```
100 FOR X=1 TO 5
110 READ Y
120 LET Z=INT(100*Y+.5)/100
130 PRINT Z
140 NEXT X
150 DATA 1.06142,27.5292,138
.021
160 DATA .423715,51.9132
170 END
```

7. N! si dice "N fattoriale" ed è il prodotto di tutti i numeri interi da 1 a N, compresi. Per esempio

3! = (1)(2)(3) = 6
5! = (1)(2)(3)(4)(5) = 120

eccetera. Scrivete un programma che riceva N in input e che poi calcoli N! e lo stampi. Provando il programma sul computer vi accorgerete che valori non troppo grandi di N danno luogo a fattoriali che sono troppo grandi per essere trattati dalla macchina. Il fattoriale di N è una funzione che cresce molto rapidamente.

8. Scrivete un programma Basic che riceva in input N voti. Fategli calcolare e stampare (1) il voto più alto, (2) il voto più basso e (3) la media dei voti.

9. C'è qualcosa di sbagliato nel seguente programma?

```
100 FOR X=1 TO 2
110 FOR Y=2 TO 6
120 PRINT X+Y
130 NEXT Y
140 FOR Z=1 TO 3
150 PRINT X+Z
160 NEXT X
170 NEXT Z
180 END
```

10. Quale sarà l'output del seguente programma, se verrà eseguito?

```
100 FOR X=1 TO 4
110 FOR Y=1 TO 3
120 LET Z=X*Y
130 PRINT Z
140 NEXT Y
150 PRINT
160 NEXT X
170 END
```

11. Supponiamo che vogliate investire un milione il primo giorno dell'anno per 10 anni ad un tasso di interesse semplice annuale del 6 per cento. Alla fine del decimo anno il valore dell'investimento sarà di 13.971.640 lire. Per calcolarlo si usa la seguente formula

```
P2=(P1+I)(1+R/100)
```

In essa, R è il tasso di interesse annuale in percentuale, I è l'investimento annuale in lire, P1 è il valore dell'investimento all'inizio di ogni anno e P2 è il suo valore alla fine dell'anno. Così P2 diventa P1 per l'anno dopo. Scrivete un programma Basic che produca, se eseguito, un output di questo tipo:

```
INVESTIMENTO ANNUALE ? 10000
00
TASSO DI INTERESSE (%) ? 8
QUANTI SONO GLI ANNI ? 20
ALLA FINE DELL'ULTIMO ANNO
IL VALORE DELL'INVESTIMENTO
SARA' 49422921.5
```

12. Le istruzioni DATA elencate sotto riportano le ore spese da un certo numero di dipendenti durante una settimana lavorativa.

```
190 DATA 5
200 DATA 2,4.8,8,10,8,7,10
201 DATA 5,3.75,7,8,8,6,10
202 DATA 1,3.25,8,10,6,8,8
203 DATA 4,5.8,10,6,10,6
204 DATA 3,4.25,6,6,8,10,7
```

Nella riga 190 è contenuto il numero dei dipendenti. Ognuna delle altre righe contiene i dati settimanali per ognuno di essi. I dati sono il numero del dipendente, la paga oraria e le ore lavorative dal lunedì al venerdì. Ai dipendenti spetta la paga di un'ora e mezza per ogni ora se hanno superato le 40 ore settimanali. Scrivete un programma Basic che, usando queste istruzioni DATA, calcoli e stampi il numero dei lavoratori e la paga totale settimanale per ognuno di essi.

13. Supponiamo che le seguenti istruzioni DATA diano i risultati di tre interrogazioni di studenti.

```
190 DATA 6
200 DATA 3,90,85,92
201 DATA 1,75,80,71
202 DATA 6,100,82,81
203 DATA 5,40,55,43
204 DATA 2,60,71,68
205 DATA 4,38,47,42
```

Nella riga 190 c'è il numero di studenti della classe. Le altre istruzioni danno i risultati dei vari studenti. I dati sono il numero di identificazione dello studente, il primo voto, il secondo voto e il terzo voto. Per esempio nella riga 202 si legge che lo studente 6 ha ottenuto i voti 100, 82 e 81. Scrivete un programma che con queste istruzioni DATA calcoli e stampi i numeri di identificazione degli studenti e i voti finali per ciascuno di essi. Si supponga che i voti delle prime due interrogazioni abbiano un peso del 25% rispetto al voto complessivo, mentre la terza valga il 50%.

14. Scrivete un programma che accetti in input una stringa di caratteri e stampi il numero di volte che ogni vocale viene ripetuta nella stringa.

15. Scrivete un programma che faccia uso dell'istruzione FOR...NEXT e che stampi tutti i 127 caratteri ASCII.

ESERCIZI

Con questi esercizi potrete provare in che misura avete capito gli argomenti del capitolo. Troverete le risposte alla fine del libro.

1. Che output otterreste eseguendo il seguente programma?

```
100 FOR Y=20 TO 1 STEP -2
110 PRINT Y,
120 NEXT Y
130 END
```

2. Cosa verrà stampato se si esegue il seguente programma?

```
100 FOR A=1 TO 4
110 FOR B=1 TO 3
120 PRINT A*B
130 NEXT B
140 NEXT A
150 END
```

3. Riempite gli spazi vuoti in maniera adeguata.

   a. SQR(36) = ________

   b. INT(7.13) = ________

   c. ABS(−22.8) = ________

   d. SGN(−1.3) = ________

4. Cosa c'è di sbagliato (eventualmente) nel seguente programma?

```
100 FOR I=1 TO 5
110 FOR J=2 TO 5
120 PRINT I,J
130 NEXT I
140 NEXT J
150 END
```

5. Le miglia possono essere tradotte in chilometri moltiplicando il loro numero per 1.609. Scrivete un programma che produca una tabella simile a questa:

Miglia	Chilometri
10	16.09
15	24.135
20	32.18
ecc.	
100	160.9

6. Dati numerici sono contenuti come segue in alcune istruzioni DATA:

```
100 DATA 10
110 DATA 25,21,24,21,26,27,2
5,24,23,24
```

Il numero della riga 100 conta i numeri passati dall'altra istruzione DATA. Scrivete un programma che usi queste istruzioni per calcolare la media dei numeri, escluso quello di riga 100.

7. Spiegate brevemente lo scopo di ciascuna delle seguenti funzioni: ABS, SGN, INT, SQR, SEG$, e VAL.

CAPITOLO OTTO

Lavorare con gruppi di informazioni

ARGOMENTI

In questo capitolo applicheremo alcune delle idee che ci siamo già fatti sui gruppi di informazioni. Verranno inoltre introdotti concetti nuovi che renderanno il nostro Basic più capace. Gli argomenti che tratteremo sono i seguenti.

Variabili stringa con indici

La nozione di variabile stringa può essere estesa alle stringhe alfanumeriche con indici. Questa possibilità consente importanti applicazioni non numeriche.

Variabili numeriche con indici

Con l'uso di variabili numeriche con indici si possono realizzare validissimi programmi che lavorano con i numeri. Per questo spiegheremo cosa intendiamo per variabili numeriche con indici e a cosa queste possono servire.

Applicazioni

Analizzeremo dei programmi in Basic, che si serviranno fondamentalmente di variabili numeriche con indici e di variabili stringa con indici.

PRATICA SUL CALCOLATORE

Conoscendo le difficoltà che i principianti di solito incontrano nell'approccio con questi argomenti, ci sembra necessario fare un'introduzione alla parte che prevede il lavoro al calcolatore.
Quando si ha a che fare con gruppi di informazioni, bisogna essere in grado di distinguere gli elementi di un gruppo tra di loro. Ecco perché si adoperano gli indici. Comunque, prima di parlare degli indici, dobbiamo aggiungere due parole importanti al nostro vocabolario di programmatori. Per descrivere un gruppo di dati potremmo usare il termine "collezione" ma di solito si usa comunemente "matrice". Per noi significherà "raccolta di dati". I singoli elementi di un gruppo di dati possono essere stringhe alfanumeriche o numeri. Per capirci meglio, consideriamo la seguente matrice unidimensionale

```
Y(1) = 9
Y(2) = 10
Y(3) = 7
Y(4) = 14
Y(5) = 12
Y(6) = 15
```

Il nome di questa matrice è Y. La sua dimensione è 6, perché essa contiene sei "elementi" o "membri". Essi sono i numeri 9, 10, 7, 14, 12 e 15. I numeri che stanno nella parentesi che segue la Y si dicono "indici". Ad ogni indice è associato un elemento della matrice. Così, Y(4) rappresenta il quarto elemento della matrice, in questo caso 14. Y(4) si legge '"Y di quattro". Il terzo numero della matrice sarà allora "Y di tre", e così via. La nostra matrice è unidimensionale, poiché basta un solo numero o indice per localizzare qualsiasi elemento.
Ora usiamo le stesse idee per considerare un esempio un po' più complicato.

```
Z$(1,1) = "CANE"   Z$(1,2) = "SU"      Z$(1,3) = "NOTA"
Z$(2,1) = "MA"     Z$(2,2) = "ROSSO"   Z$(2,3) = "NON"
```

In questo caso la matrice di stringhe di caratteri Z contiene sei elementi.

Poiché si tratta di una matrice di stringhe alfanumeriche, i suoi elementi sono parole. Questa volta, però, si tratta di una matrice bidimensionale (o vettore), perché per localizzare un elemento dobbiamo specificare la riga e la colonna in cui si trova. Il primo indice dà il numero della riga, il secondo specifica la colonna. Z(2,1) si legge "Z di due uno" e rappresenta l'elemento Z che si trova nella seconda riga e nella prima colonna. Nel nostro esempio, Z$(2,1) è la parola MA. Analogamente, Z$(1,3) è NOTA, e così via. Sul TI home computer possiamo anche avere una matrice tridimensionale. Il concetto è un'estensione delle matrici uni e bidimensionali. In questo caso abbiamo numeri di riga, colonna e "pagina". Ad esempio, A(2,3,5) rappresenta l'elemento numerico della matrice A posto alla riga 2, alla colonna 3 e a pagina 5. Similmente, T$(1,4,2) identifica la stringa di caratteri della matrice T$ di riga 1, colonna 4, pagina 2.
Riassumendo avremo a che fare con tre tipi di vettori. I vettori unidimensionali necessitano di un singolo indice per localizzare ogni elemento. I vettori bidimensionali, che chiameremo matrici, hanno bisogno di due indici per ogni elemento (numero di riga e numero di colonna). Le matrici tridimensionali hanno bisogno di tre indici per ogni elemento (numero di riga, di colonna e di pagina). Sia i vettori che le matrici possono essere numerici o alfanumerici. Al vettore è associata l'idea di variabile ad un indice. Allo stesso modo le variabili a due indici vengono usate nelle matrici bidimensionali e quelle a tre indici nelle matrici tridimensionali. Letta questa breve introduzione potete passare al lavoro sul computer.

1. Accendete il vostro calcolatore e scrivete il seguente programma.

```
100 LET A$(1)="CASA"
110 LET A$(2)="STALLA"
120 LET A$(3)="BARACCA"
130 LET A$(4)="DEPOSITO"
140 LET A$(5)="CAPANNA"
150 PRINT A$(4)
160 END
```

Quale pensate che sarà il risultato se eseguiamo il programma?

___

Fatelo funzionare e riportate cosa è successo.

___

2. Bene, cambiate la riga 150 con

```
150 PRINT A$(1),A$(3)
```

Cosa pensate che accadrà?

Eseguite il programma e copiate ciò che il computer ha stampato.

---

3. Sostituite nella riga 150 la virgola con un &, cosicché risulti

```
150 PRINT A$(1)&A$(3)
```

Eseguite il programma e segnate cosa è successo.

Qual è l'effetto del & nella stampa di stringhe di caratteri?

---

4. Cancellate il programma dalla memoria e azzerate il video. Scrivete questo programma:

```
100 FOR I=1 TO 5
110 READ B$(I)
120 NEXT I
130 DATA "ROSSO","BIANCO","B
LU"
140 DATA "VERDE","MARRONE"
150 PRINT B$(3)
160 END
```

Esaminate un pò' il programma. Quale sarà secondo voi il risultato in esecuzione?

Fatelo funzionare e controllate se avete sbagliato.

---

5. Cancellate le righe 150 e 160 del programma. Aggiungete invece queste altre:

```
150 FOR I=1 TO 5
160 PRINT B$(I),
170 NEXT I
180 END
```

E ora cosa pensate che accadrà?

Eseguite il programma e segnate ciò che il computer ha fatto.

---

6. Sostituite alla riga 150 la seguente

```
150 FOR I=5 TO 1 STEP -2
```

Eseguite il programma e trascrivete il risultato.

---

7. Estendiamo un po' il nostro argomento. Cancellate il programma che è in memoria e scrivete il seguente:

```
100 LET C$(1,1)="BIANCO"
110 LET C$(1,2)="NERO"
120 LET C$(1,3)="MARRONE"
130 LET C$(2,1)="AUTOMOBILE"
140 LET C$(2,2)="BICICLETTA"
150 LET C$(2,3)="AEROPLANO"
160 FOR I=1 TO 2
170 PRINT C$(I,2)
180 NEXT I
190 END
```

Questo programma è più complesso del precedente, ma dovreste essere in grado di riconoscere qual è il suo scopo. Fatelo funzionare e riportate quel che avviene.

---

8. Bene, modificate così la riga 170

```
170 PRINT C$(I,3)
```

Che output si otterrà ora?

Eseguite il programma e prendete nota di ciò che è successo.

---

9. Cambiate le righe 160, 170 e 180 con

```
160 FOR J=1 TO 3
170 PRINT C$(1,J)
180 NEXT J
```

Che scopo ha ora il programma?

Eseguitelo e trascrivete l'output ottenuto.

---

10. Sostituite la riga 170 con questa:

```
170 PRINT C$(2,J)
```

Cosa si otterrà?

Fate funzionare il programma e segnate ciò che è accaduto.

---

11. Finora abbiamo lavorato con gruppi di parole. Lo stesso si può fare con i numeri. Cancellate il programma dalla memoria e scrivete quest'altro:

```
100 LET X(1)=21
110 LET X(2)=13
120 LET X(3)=16
130 LET X(4)=8
140 LET X(5)=11
150 PRINT X(1)
160 END
```

Cosa supponete che succeda, eseguendolo?

Fatelo e riportate ciò che è successo.

---

12. Adesso modificate il programma per fargli stampare il quarto valore di X. Eseguite il programma. Funziona?

---

13. Bene, cambiate la riga 150 con la seguente:

```
150 PRINT X(3)+X(4)
```

Fate una lista del programma ed analizzatela brevemente.
Cosa pensate che accadrà se lo eseguite?

Fatelo e accertatevi di aver detto giusto. Riportate sotto ciò che in effetti è stato stampato.

---

14. Scrivete

```
150 FOR I=1 TO 5
152 PRINT X(I)
154 NEXT I
```

Listate, il programma. Cosa pensate che stamperà?

Guardate se avete risposto correttamente. Trascrivete sotto cosa è successo in esecuzione.

---

15. Modificate questo programma in modo che dia in output solo i primi tre valori del vettore X. Dopo aver provato segnate ciò che avete ottenuto.

---

16. Cambiate ancora il programma perché faccia stampare il primo valore del vettore, seguito da tutti gli altri. Annotate sotto quel che succede.

---

17. Cancellate il programma che è contenuto in memoria. Scrivete ora il seguente.

```
100 LET Y(1,1)=2
110 LET Y(1,2)=5
120 LET Y(1,3)=1
130 LET Y(2,1)=2
140 LET Y(2,2)=4
150 LET Y(2,3)=3
160 PRINT Y(1,3)
170 END
```

Fatene una lista ed assicuratevi di averlo copiato correttamente. Cosa pensate che faccia?

Eseguite il programma e prendete nota di quello che è stato stampato.

---

18. Scrivete

```
160 PRINT Y(2,2)+Y(1,3)+Y(1,
1)
```

Ora listate il programma. Qual è il suo scopo?

Fatelo funzionare e accertatevi di aver risposto correttamente.

---

19. Scrivete

```
160 LET S=0
162 FOR J=1 TO 3
164 LET S=S+Y(1,J)
166 NEXT J
168 PRINT S
```

Fate una lista del programma ed esaminatelo attentamente. Che risultato otterremo eseguendolo?

Fatelo e segnate quello che è risultato.

Spiegate con le vostre parole ciò che avviene all'interno del programma.

20. Scrivete

```
162 FOR I=1 TO 2
164 LET S=S+Y(I,2)
166 NEXT I
```

Listate il programma. Cosa fa, secondo voi?

Eseguitelo e riportate sotto quel che viene stampato.

Cercate di esporre con le vostre parole anche in questo caso come funziona il tutto.

21. Ora scrivete

```
162 FOR I=1 TO 2
164 FOR J=1 TO 3
166 LET S=S+Y(I,J)
168 NEXT J
170 NEXT I
172 PRINT S
180 END
```

Fate una lista e riflettete per un po' sul programma. In particolare confrontatelo con quelli dei passi 19 e 20.
Cosa fa questo programma?

Fatelo funzionare e scrivete ciò che viene stampato.

---

22. Cancellate il programma in memoria. Scrivete invece quest'altro:

```
100 DIM X(12),Y(12)
110 FOR I=1 TO 12
120 READ X(I),Y(I)
130 NEXT I
140 PRINT X(1)+Y(4)
150 DATA 2,1
151 DATA -1,3
152 DATA 5,6
153 DATA 2,4
154 DATA 3,1
155 DATA 8,4
156 DATA 5,1
157 DATA 3,4
158 DATA 6,2
159 DATA 1,1
160 DATA 7,7
161 DATA 5,3
170 END
```

Fatene una lista ed accertatevi di averlo introdotto correttamente. Esaminatelo con attenzione. Eseguendolo, cosa si otterrà?

Fatelo funzionare e guardate se avevate ragione. Trascrivete sotto i risultati.

---

23. Scrivete

```
100
```

e fate una lista completa. Cosa è successo?

Eseguite il programma e prendete nota di ciò che è avvenuto.

Vi sembra che l'istruzione DIM che era presente originariamente nel programma fosse necessaria?

---

24. Scrivete

```
100 DIM X(9),Y(9)
110 FOR I=1 TO 9
```

Listate il programma. Cosa accadrà ora se lo eseguiamo?

Provateci e controllate se avete risposto correttamente.

---

25. Scrivete

```
100
```

Facendo così avete cancellato la riga 100 dal programma. Funzionerà anche se è stata eliminata l'istruzione DIM?

Provateci e trascrivete il risultato.

Confrontate gli output dei passi 23 e 25. In certi casi l'istruzione DIM è necessaria, mentre in altri no. Torneremo più tardi su questa questione.

26. Cancellate il programma dalla memoria. Inserite ora il seguente.

```
100 DIM A(4,3)
110 FOR I=1 TO 4
120 FOR J=1 TO 3
130 READ A(I,J)
140 NEXT J
150 NEXT I
160 FOR I=1 TO 4
170 FOR J=1 TO 3
180 PRINT A(I,J);
190 NEXT J
200 PRINT
210 PRINT
220 NEXT I
230 DATA 1,3,1
240 DATA 4,2,5
250 DATA 1,4,2
260 DATA 3,2,5
270 END
```

Assicuratevi di averlo scritto correttamente ed esaminatelo per qualche minuto. Riuscite a vedere cosa si otterrà in esecuzione?

Eseguite il programma e annotate il risultato.

Confrontate l'output con i numeri contenuti nelle istruzioni DATA del programma.

27. Ora che avete visto l'uso di vettori e matrici bidimensionali, consideriamo brevemente una matrice a tre dimensioni. Cancellate la memoria ed inserite il seguente programma.

```
100 DIM A(2,3,2)
110 FOR P=1 TO 2
120 FOR R=1 TO 2
130 FOR C=1 TO 3
140 READ A(R,C,P)
150 NEXT C
160 NEXT R
170 NEXT P
180 REM PAGINA 1
190 DATA 5,3,6
200 DATA 2,1,2
210 REM PAGINA 2
220 DATA 3,4,3
230 DATA 1,5,1
240 PRINT A(1,1,1)+A(2,1,2)
250 END
```

Questo programma sembra complicato ma dovreste essere già in grado di comprendere come funziona. In particolare, concentrate l'attenzione sul concetto di riga, colonna e pagina implicito nella variabile con indici A(R,C,P) di linea 140. Cosa pensate apparirà se avvierete il programma?

Eseguite il programma ed osservate cosa avviene.

---

28. Ora apportate i seguenti cambiamenti.

```
250 LET S=0
260 LET P=1
270 FOR R=1 TO 2
280 FOR C=1 TO 3
290 LET S=S+A(R,C,P)
300 NEXT C
310 NEXT R
320 PRINT S
330 END
```

Cosa succederà ora se eseguiremo il programma?

Provate ed annotate i risultati.

---

29. Ora cambiate la linea 260 e ponete P uguale a 2. Cosa avverrà eseguendo ora il programma?

Avviate il programma e riportate qui sotto ciò che appare sul video.

---

30. Per procedere oltre sarà necessario che colleghiate al computer un registratore a cassette. Se non ne avete uno passate al paragrafo successivo.

31. Annullate la memoria ed inserite il seguente programma.

```
100 OPEN #1:"CS1",OUTPUT,FIX
ED 64
110 FOR I=1 TO 3
```

```
120 READ A$,N
130 PRINT #1:A$
140 PRINT #1:N
150 NEXT I
160 CLOSE #1
170 DATA "ALBERTO",215
180 DATA "MARIA",142
190 DATA "GIACOMO",193
200 END
```

Questo programma presenta alcune particolarità che non avevate visto in precedenza. Precisamente, le istruzioni OPEN e CLOSE alle linee 100 e 160, così pure la forma differente assunta dall'istruzione PRINT nelle linee 130 e 140. Assicuratevi di aver connesso correttamente il registratore al computer e che vi sia inserita una cassetta non ancora utilizzata. Eseguite il programma. Cosa è successo?

---

32. Bene, seguite le istruzioni che appaiono sullo schermo e poi premete il tasto ENTER. Cosa fa il computer?

Continuate a seguire le istruzioni e poi premete ENTER. Cosa accade sul registratore?

---

33. A questo punto dovreste vedere che il nastro magnetico si sta avvolgendo. Come probabilmente avete supposto, in questo momento i dati vengono scritti sul nastro. Cosa appare quando il nastro si ferma?

---

34. Seguite le istruzioni che si presentano sullo schermo e al termine togliete il nastro dal registratore. Ora eseguiamo il processo inverso e leggiamo i dati dal nastro introducendoli nuovamente nel computer per ottenerne la stampa. Cancellate la memoria ed inserite il seguente programma.

```
100 OPEN #1:"CS1",INPUT ,FIX
ED 64
110 FOR I=1 TO 3
120 INPUT #1:A$
130 INPUT #1:N
140 PRINT A$,N
150 NEXT I
160 CLOSE #1
170 END
```

Studiate il listato per qualche istante e confrontatelo con quello precedente per mettere in luce somiglianze e differenze.

35. Ora eseguite il programma seguendo le istruzioni che appaiono di volta in volta sullo schermo. Cosa accade al termine?

___

36. Ciò conclude per ora l'attività pratica sul computer. Spegnete il calcolatore e passate alla discussione.

DISCUSSIONE

È naturale a questo punto che l'argomento dei vettori, siano essi numerici o alfanumerici, risulti un po' confuso. Perciò è importante che prestiate particolare attenzione a questa fase di discussione, in modo da poter risolvere qualunque dubbio vi sia venuto lavorando col computer.

Le variabili con indice

Quando si ha a che fare con un gran numero di informazioni l'uso di variabili con indice diventa indispensabile. Non importa se i dati sono numerici o alfanumerici. Se, ad esempio, dovessimo scrivere un programma che contiene solo quattro numeri, non avremmo nessuna difficoltà ad identificarli. Potremmo chiamarli X, Y, U e V. Ma se i numeri fossero 100? Per questo, e per altre ragioni, le variabili con indice sono così utili. Il Basic, per fortuna, dispone di indici che possono essere associati sia alle variabili stringa che alle variabili numeriche, purché pronte per essere usate.
Consideriamo il seguente insieme di dati numerici.

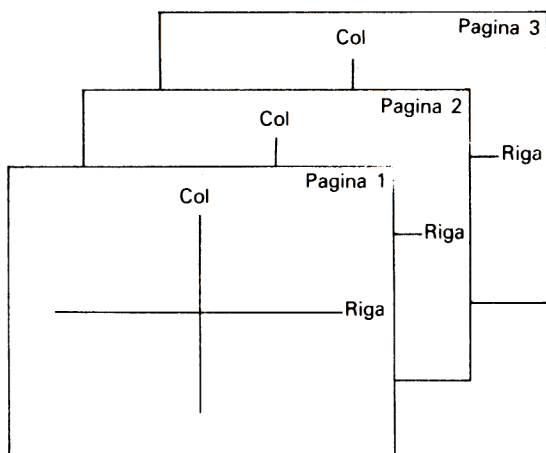
i	Y_i
1	14
2	8
3	9
4	11
5	16
6	20
7	5
8	3

Possiamo chiamare Y l'insieme di tutti questi numeri. Così Y è una "collezione di numeri" o "vettore", per noi sarà lo stesso. Per localizzare un numero in un vettore dobbiamo conoscere il nome di quest'ultimo (in questo caso Y) e la posizione di quel numero all'interno di esso. Per questo abbiamo scritto la colonna delle i. Quindi Y(3) rappresenta il terzo numero nel vettore Y. In questo caso Y(3) ha valore 9. Allo stesso modo Y(7) è 5, Y(1) è 14 e così via. In generale parleremo di Y(i), che denota un qualsiasi elemento del vettore, a seconda dell'indice i. Se i fosse 8 nel nostro esempio Y(i) sarebbe 3. Poiché abbiamo usato un solo indice per localizzare un qualsiasi elemento del vettore, la nostra raccolta di numeri è unidimensionale. Consideriamo ora una matrice numerica bidimensionale.

$Y_{i,j}$	1	2	3	4
1	3	−1	10	8
2	2	4	5	6
3	1	−2	9	3

Ora sono necessari due numeri per specificare la posizione di un elemento qualunque. Essa è completamente determinata da un numero di riga e un numero di colonna. Per esempio, Y(1,3) rappresenta l'elemento di Y che sta nella prima riga e nella terza colonna. Se guardiamo la tabella, troviamo il valore 10. Un elemento generico di una matrice Y si denota con Y(I,J). Il primo indice (I) è il numero di riga e il secondo (J) è il numero di colonna.
Per essere sicuri di come effettivamente si usano gli indici doppi, controllate le seguenti espressioni nella tabella sopra.

$$Y_{3,2} = -2$$
$$Y_{1,4} = 8$$
$$Y_{3,3} = 9$$
$$Y_{2,1} = 2$$

Estendendo i concetti precedenti ad un'ulteriore dimensione si ottiene una matrice tridimensionale. In questo caso il terzo indice indica il numero di "pagina".
Va specificato che il termine "pagina" non è propriamente corretto ma è soltanto un modo conveniente per raffigurare mentalmente il terzo indice. Il seguente diagramma mostra come sono organizzati gli indici.

Allora per individuare un elemento dobbiamo specificare in quale pagina si trova e i suoi numeri di riga e di colonna. Usando la notazione X(5,3,2) intendiamo l'elemento della matrice numerica X che si trova a riga 5, colonna 3 e pagina 2.

Nel linguaggio Basic gli indici vengono scritti in una parentesi che segue il nome del vettore o della matrice Quindi Y(2) non è la moltiplicazione di Y per 2, ma il secondo elemento del vettore Y. B$(5,8) significa "B$ di 5 e di 8". A questo punto potremo chiederci se l'espressione X(M−N+3,S*T) ha qualche significato. La risposta è affermativa dato che il computer converte automaticamente M-N+3 e S*T in numeri maggiori o uguali a zero. Ma bisogna tenere presente una cosa importante. Supponiamo di voler cercare X(A+B), dove A = 2.6 e B = 1.1. Allora A+B = 3.7, ma non ha alcun senso cercare il 3.7esimo numero del vettore X. Il calcolatore perciò arrotonderà il numero all'intero più vicino, in questo caso X(A+B) risulterà X(4) cioè il quarto elemento di X.

Tutto ciò che si è detto sui vettori numerici vale anche per i vettori con stringhe di caratteri. Siccome adesso avete abbastanza familiarità con questo concetto, d'ora in poi tralasceremo "di caratteri" e ci riferiremo ad una collezione di caratteri chiamando "vettore di stringhe" e "variabile stringa" rispettivamente un "vettore di stringhe di caratteri" e una "variabile stringa alfanumerica".

Ecco un esempio di vettore di stringhe:

```
X$(1)="FIGLIO"
X$(2)="FIGLIA"
X$(3)="MADRE"
X$(4)="PADRE"
X$(5)="ZIO"
X$(6)="ZIA"
```

Le parole costituiscono gli elementi del vettore. I numeri da 1 a 6 sono gli indici che identificano le varie parole. Il computer, per quanto riguarda gli indici, tratta i vettori di stringhe come i vettori numerici. Questo è invece un esempio di vettore di stringhe a due dimensioni

```
A$(1,1)="AA"   A$(1,2)="AB"
A$(2,1)="BA"   A$(2,2)="BB"
A$(3,1)="CA"   A$(3,2)="CB"
```

In questo caso gli elementi sono coppie di caratteri, per mostrare che gli elementi sono proprio gruppi di caratteri. Non è necessario che siano parole con qualche significato.
Ed ora un ultimo commento sulle variabili stringa. Esse possono essere lette dalle istruzioni DATA così come vengono lette le variabili numeriche. Comunque, se usate le stringhe nelle istruzioni DATA, racchiudetele tra virgolette, per essere sicuri. Se l'istruzione READ contiene sia variabili numeriche che variabili stringa, bisogna stare attenti che le informazioni delle istruzioni DATA vengano associate a variabili di quello stesso tipo. Se, ad esempio, il calcolatore richiede una variabile numerica e l'istruzione corrispondente delle istruzioni DATA è una variabile stringa, il calcolatore si ferma e segnala un errore.

Creare uno spazio nella memoria per le matrici

Prima di discutere l'istruzione DIM (*dimension*), dobbiamo considerare più da vicino la nozione di indice in una matrice. In particolare ci si potrebbe chiedere "qual è l'indice minore possibile?"
Nelle attività pratiche il problema non era stato sollevato e probabilmente avevate assunto tacitamente che il minor indice possibile fosse uno. Alcuni computer usano uno come indice minore mentre altri usano lo zero. Il TI home computer consente entrambe le soluzioni! Questo si può ottenere con l'uso dell'istruzione OPTION. Quando accendete il computer l'indice più basso (base) è lo zero. La base era zero anche durante la pratica al computer ma siete stati deliberatamente tenuti lontani da situazioni dove il fatto potesse essere notato.
Per cambiare la base a 1 dovete inserire l'istruzione OPTION BASE 1 nel programma. E ciò va fatto all'inizio prima che venga fatto ogni riferimento alla matrice. In un programma può esserci solo un'istruzione OPTION. Per evitare confusioni, probabilmente sarebbe opportuno includere o l'istruzione OPTION BASE 1 o OPTION BASE 0 in tutti i programmi che fanno uso di matrici. In queto modo non potranno esserci dubbi sulla base degli indici delle matrici presenti nel programma.
Ci sono circostanze in cui la base zero per gli indici è necessaria. Tuttavia, se non c'è alcuna esigenza specifica per la base zero, è buona norma

dichiarare OPTION BASE 1, anche perché in questo modo le matrici occupano meno spazio nella memoria.
Il computer deve conoscere quanto è grande una matrice per due ragioni. Per prima cosa, bisogna riservare spazio in memoria per contenerla. In secondo luogo, la grandezza della matrice deve essere nota per poter svolgere adeguatamente le operazioni su di essa.
In effetti, se i vettori sono piccoli, il Basic predispone automaticamente lo spazio per contenerli. Se si usa un vettore ad una dimensione, il Basic automaticamente predispone lo spazio per dieci elementi (OPTION BASE 1) oppure per undici elementi (OPTION BASE 0) anche in assenza dell'istruzione DIM. Quando invece si usa una matrice a due dimensioni e l'istruzione DIM non è presente, il Basic predispone lo spazio per una matrice dieci per dieci o undici per undici a seconda della base scelta. La stessa cosa vale per le matrici tridimensionali e, a seconda della base, viene ricavato lo spazio per una matrice dieci per dieci per dieci o undici per undici per undici. Ma questa caratteristica del Basic non è molto utile. Consigliamo infatti l'uso delle istruzioni di dimensionamento in tutti i programmi, qualunque sia la grandezza dei vettori e delle matrici. La correzione di un programma che usa vettori o matrici è resa problematica dall'assenza dell'istruzione DIM.
Ecco un esempio di istruzione DIM (che sta per "dimensione")

```
100 DIM B(5,20),Y(3,4,6),Z(3
4),X$(3,6)
```

Nella riga 100 abbiamo dimensionato un vettore e tre matrici. B è una matrice numerica con cinque righe e venti colonne. Y è una matrice numerica tridimensionale con tre righe, quattro colonne e sei pagine, Z è un vettore numerico con trentaquattro elementi ed infine X$ è una matrice di stringhe con tre righe e sei colonne. È bene che le istruzioni DIM e OPTION siano le prime del programma. Ciò rende possibile una veloce valutazione dello spazio di memoria che i vettori del programma richiedono. Comunque, le istruzioni DIM e OPTION devono sicuramente precedere ogni altra istruzione che contenga vettori o matrici. Come abbiamo detto già, anche quando il Basic non le richiede, è utile usare le istruzioni di dimensionamento.

Variabili con indici e cicli FOR...NEXT

Poiché gli indici rappresentano elementi di un gruppo di dati e le operazioni con gruppi di dati sono quasi sempre di tipo ripetitivo, è ragionevole, quando si ha a che fare con matrici o vettori, usare le istruzioni FOR...NEXT. Ad esempio, questo pezzo di programma costruisce una matrice sei per quattro e la riempie di 5.

```
100 DIM A(6,4)
110 OPTION BASE 1
120 FOR R=1 TO 6
130 FOR C=1 TO 4
140 LET A(R,C)=5
150 NEXT C
160 NEXT R
```

Se analizziamo in dettaglio il programma, tutto appare chiaro. Quando si incontra per la prima volta la riga 140, R = 1 e C = 1. Poi R rimane costante, mentre C assume i valori 2, 3 e 4. Ad ognuno di questi passi un elemento del vettore è posto uguale a 5. Poi R diventa 2 e C assume uno dopo l'altro i valori 1, 2, 3 e 4. Il procedimento continua finché tutti gli elementi sono posti uguali a 5. Sia i vettori che le matrici bi- e tridimensionali si possono trattare in questo modo. I cicli, le matrici ed i vettori aumentano le capacità del computer e sono mezzi potenti per il programmatore.

Scrivere informazioni su file

Nelle attività pratiche avete incontrato un esempio nel quale i dati (stringhe e numeri) erano scritti su un nastro per registratore. Qualsiasi uso serio del computer normalmente implica la memorizzazione di dati sotto forma di file. Saper memorizzare tali dati su un nastro magnetico per poterli poi richiamare in qualsiasi momento è fondamentale in quasi tutte le applicazioni di gestione delle informazioni. Ora prenderemo in considerazione nei dettagli le procedure per memorizzare dati su un nastro magnetico.
In un programma volto alla creazione di record per prima cosa dobbiamo preoccuparci di aprire la comunicazione con l'unità di registrazione. Questo viene fatto con l'istruzione OPEN, della quale qui sotto viene dato un esempio.

```
100 OPEN #1:"CS1",OUTPUT,FIX
ED 64
```

In questa istruzione, 1 si riferisce al canale di comunicazione attraverso il quale i dati vengono trasferiti al registratore. Questo numero può essere qualsiasi intero compreso tra 1 e 255. L'unica ragione per usare più di un canale sussiste quando il computer comunica con più di un dispositivo contemporaneamente. Poiché limiteremo il nostro lavoro all'uso di un singolo registratore selezioneremo sempre il canale 1.
I caratteri racchiusi tra virgolette, dopo il numero del canale sono il nome del file nel quale verranno scritti i dati. In questo caso CS1 indica che verrà usato il registratore numero uno. Poi vengono specificate le caratteristiche del file.

Poiché il nostro primo obbiettivo è di memorizzare dati ed informazioni su un nastro magnetico, specifichiamo che il file è di tipo OUTPUT. Infine, FIXED 64 indica che le informazioni verranno trasmesse in blocchi (o record) di lunghezza costante di sessantaquattro caratteri. È importante comprendere che se, ad esempio, volessimo memorizzare una parola composta da quindici caratteri, il computer creerà ancora record di sessantaquattro caratteri completando con spazi vuoti la porzione non utilizzata del blocco di lunghezza fissata.
L'istruzioni OPEN è il presupposto necessario per inviare i dati dal computer all'unità di registrazione. Finché avrete a che fare con un unico dispositivo di output (il registratore) e vi limiterete a record di lunghezza costante uguale a sessantaquattro caratteri, potrete sempre adoperare l'istruzione considerata in precedenza.
Per inviare le informazioni si usa l'istruzione PRINT. Ad esempio

```
200 PRINT #1:X$
```

Notate che questa PRINT differisce dalle altre usate prima poiché in essa viene specificato il numero del canale attraverso il quale passano le informazioni. Il numero deve essere lo stesso di quello usato nell'istruzione OPEN. Noi useremo sempre il numero uno. In questo esempio di istruzione PRINT la stringa X$ verrà trasferita attraverso il canale numero 1 per essere memorizzata. Naturalmente, se volessimo, potremmo trattare allo stesso modo variabili numeriche. Non ha alcuna importanza da dove le informazioni provengono e da dove vengono generate. Con il comando PRINT vengono inviate al registratore. Un'ultima considerazione è che sarà tutto più semplice se in ogni istruzione PRINT verrà data solo una quantità (sia essa una stringa o un numero).
Quando tutte le informazioni sono state inviate all'unità di registrazione dobbiamo interrompere la via di comunicazione. Per questo si adopera la seguente istruzione:

```
300 CLOSE #1
```

Come probabilmente vi aspettavate, dobbiamo specificare il canale che stiamo chiudendo. Poiché abbiamo stabilito di usare solo un canale alla volta, ci possiamo riferire ad esso sempre con il numero uno.
Tutti i programmi volti al mantenimento di dati sotto forma di file su un nastro magnetico hanno la seguente struttura:

```
100 OPEN #1:"CS1",OUTPUT,FIXED 64
                    .
                    .
                    .
(inserimento dati da registrare)
                    .
                    .
                    .
500 PRINT #1: (stringa o numeri)
                    .
                    .
                    .
800 CLOSE #1
900 END
```

Il programma deve tornare ciclicamente all'istruzione PRINT finché tutti i dati non saranno stati trasmessi. Prima di memorizzare i dati veri e propri è buona consuetudine fare in modo che il primo record contenga il numero dei record che verranno poi trasmessi. In questo modo, quando i dati verranno richiamati dal computer, il programma leggerà per prima cosa questo numero e così verrà a conoscenza di quante letture successive dovrà compiere.
Quando viene eseguito un programma, se c'è un'istruzione OPEN, compare sullo schermo il seguente messaggio

```
* REWIND CASSETTE TAPE CS1
  THEN PRESS ENTER
```

Successivamente, se (come in questo caso) l'istruzione OPEN è verso un file in modo output, si vedrà la scritta

```
* PRESS CASSETTE RECORD CS1
  THEN PRESS ENTER
```

Naturalmente, in entrambi i casi, dovete premere il tasto ENTER dopo aver eseguito le operazioni richieste. A questo punto i dati possono essere scritti sul nastro.
Quando giunge l'istruzione CLOSE, il computer invia la scritta

```
* PRESS CASSETTE STOP CS1
  THEN PRESS ENTER
```

Portate a termine le istruzioni, il processo di registrazione è completato. Creare un file su un nastro magnetico non presenta alcuna difficoltà. Ricordate di aprire la comunicazione con l'istruzione OPEN (in modo output) e di interrompere la comunicazione con l'istruzione CLOSE. Le informazioni vengono inviate al file con l'istruzione PRINT. Tutte e tre queste istruzioni devono portare lo stesso numero di file, per non sba-

gliare usate sempre il numero uno. Dopo aver dato il via all'esecuzione del programma, seguite le istruzioni che appariranno sullo schermo.

Leggere le informazioni da un file

Dopo aver scritto (o memorizzato) le informazioni con un registratore sotto forma di file, dobbiamo essere in grado di scrivere un programma per richiamare gli stessi dati e ricondurli dal file al computer. Tutti i programmi volti a questo compito hanno la stessa forma generale.

```
100 OPEN #1:"CS1",INPUT,FIX
ED 64
                    .
                    .
                    .
300 INPUT #1:  (stringa o numeri)
                    .
                    .
                    .
500 CLOSE #1
600 END
```

Questa struttura è la stessa del programma per memorizzare i dati. L'istruzione OPEN ha lo stesso scopo che aveva in precedenza solo che ora il file è in INPUT. Dobbiamo ripetere un ciclo che contenga un'istruzione INPUT tante volte quante sono necessarie per trasferire tutti i dati dal nastro magnetico. L'istruzione CLOSE interrompe la comunicazione con il registratore come in precedenza. In queste tre istruzioni useremo sempre il numero di file uno.
Si rendono ora necessarie alcune considerazioni. Per prima cosa, generalmente, bisogna essere sicuri che il primo blocco di informazioni del file contenga il numero di record che sono stati memorizzati successivamente. Allora, la lettura di questa quantità, in un programma strutturato per introdurre i dati da un registratore a cassette, servirà perché venga chiesto il giusto numero di blocchi di informazioni.
In secondo luogo l'input dei dati deve essere conforme al modo in cui sono stati scritti. Se il programma chiede l'introduzione di una stringa, l'informazione presente sul nastro a quel punto deve essere una stringa. Analogamente se è richiesto l'input di una variabile numerica, il dato successivo sul nastro deve essere un numero. Infine, in ogni istruzione INPUT sarebbe preferibile chiedere l'inserimento di una singola quantità (sia essa una stringa o un numero).
Quando si esegue un programma per l'input di dati da un nastro magnetico, l'unica differenza è che nelle istruzioni apparirà il messaggio

```
* PRESSE CASSETTE PLAY CS1
  THEN PRESS ENTER
```

Quando scrivete i vostri dati in un file o leggete informazioni da un file, inizialmente usate programmi di modeste proporzioni. Quando avrete raggiunto una maggiore familiarità con il procedimento e capito chiaramente cosa avviene, allora potrete intraprendere la stesura di più ambiziosi programmi per la gestione dei dati.

ESEMPI DI PROGRAMMI

L'uso delle variabili con indici permette di trattare facilmente in Basic molti problemi interessanti. Ora prenderemo in considerazione alcuni programmi per mostrare come tali problemi vadano affrontati.

Esempio 1 - I voti dell'interrogazione

Per illustrare il concetto di vettore consideriamo un problema che sta a cuore a molta gente: quello dei voti di una interrogazione. Supponiamo che i quindici studenti di una classe, ad una interrogazione, abbiamo ottenuto i seguenti risultati (in centesimi).

Numero dello studente

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Voto	67	82	94	75	48	64	89	91	74	71	65	83	72	69	72

Il problema consiste nello scrivere un programma Basic che richieda i voti. Il risultato dovrà essere di questo tipo:

```
QUANTI SONO GLI STUDENTI? 15
STUDENTE              VOTO

1                     ? 67
2                     ? 82
3                     ? 94
4                     ? 75
5                     ? 48
6                     ? 64
7                     ? 89
8                     ? 91
9                     ? 74
10                    ? 71
11                    ? 65
12                    ? 83
13                    ? 72
14                    ? 69
15                    ? 72
```

Il programma dovrebbe poi calcolare la media della classe, cercare il voto più alto e quello più basso, e stampare questi risultati così:

```
LA MEDIA DELLA CLASSE E' 74.4
IL VOTO PIU' ALTO E' 94
IL VOTO PIU' BASSO E' 48
```

Come negli altri esercizi, procediamo passo per passo. Per prima cosa, siccome raccoglieremo i voti degli studenti usando le variabili con indici, dobbiamo scrivere le istruzioni OPTION BASE e DIM per riservare spazio al vettore.

```
100 OPTION BASE 1
110 DIM G(50)
```

Usiamo la variabile G per raccogliere i voti, che possono essere al massimo cinquanta. Facciamo stampare la domanda, poi richiediamo un input e saltiamo una riga.

```
120 PRINT "QUANTI SONO GLI S
TUDENTI";
130 INPUT N
140 PRINT
```

Ora possiamo inserire i voti. Prima però dobbiamo scrivere l'intestazione della tabella.

```
150 PRINT "STUDENTE","VOTO"
160 PRINT
```

L'ideale per eseguire l'input dei voti è un ciclo FOR ... NEXT.

```
170 FOR I=1 TO N
180 PRINT I,
190 INPUT G(I)
200 NEXT I
```

Il numero di ogni studente viene stampato nella riga 180. Nella linea 190 lo stesso numero (I) viene usato come indice per il voto. Questi sono identificati dal computer come G(1), G(2), ..., G(N). Il prossimo obbiettivo consiste nel calcolare la media dei voti. Ciò si fa sommando tutti i voti e dividendo per il loro numero.

```
210 LET S=0
220 FOR I=1 TO N
230 LET S=S+G(I)
240 NEXT I
250 PRINT
```

Ora calcoliamo la media e stampiamo i risultati.

```
260 LET M=S/N
270 PRINT "LA MEDIA DELLA CL
ASSE E'";M
```

La parte finale del programma contiene la ricerca e la stampa del voto più alto e di quello più basso, che saranno rappresentati, rispettivamente, da H e da L. All'inizio poniamo H ed L uguali al primo voto della lista, cioè a G(1). Sappiamo che un voto non può essere al tempo stesso il più alto e il più basso; così passeremo in rassegna tutti gli altri voti, li confronteremo con H ed L e sostituiremo i valori di H ed L quando sarà il caso.

```
280 LET H=G(1)
290 LET L=G(1)
300 FOR I=2 TO N
310 IF L<G(I)THEN 330
320 LET L=G(I)
330 IF H>G(I)THEN 350
340 LET H=G(I)
350 NEXT I
```

La stampa si ottiene con due sole righe.

```
360 PRINT "IL VOTO PIU' ALTO
 E'";H
370 PRINT "IL VOTO PIU' BASS
O E'";L
```

L'istruzione END chiude il programma.

```
380 END
```

Ecco il programma completo:

```
100 OPTION BASE 1
110 DIM G(50)
120 PRINT "QUANTI SONO GLI S
TUDENTI";
130 INPUT N
140 PRINT
150 PRINT "STUDENTE","VOTO"
160 PRINT
170 FOR I=1 TO N
180 PRINT I,
190 INPUT G(I)
200 NEXT I
210 LET S=0
220 FOR I=1 TO N
230 LET S=S+G(I)
240 NEXT I
250 PRINT
260 LET M=S/N
270 PRINT "LA MEDIA DELLA CL
ASSE E'";M
280 LET H=G(1)
290 LET L=G(1)
300 FOR I=2 TO N
310 IF L<G(I)THEN 330
320 LET L=G(I)
330 IF H>G(I)THEN 350
340 LET H=G(I)
350 NEXT I
360 PRINT "IL VOTO PIU' ALTO
 E'";H
370 PRINT "IL VOTO PIU' BASS
O E'";L
380 END
```

Eseguite questo programma con il vostro computer usando i dati della tabella iniziale. Se avete qualche problema con la ricerca del massimo e del minimo voto nelle righe da 280 a 350, analizzate il programma dettagliatamente.

Esempio 2 - I voti di una classe

Possiamo estendere facilmente le idee dell'Esempio 1 ad una matrice bidimensionale. Supponiamo, ora, di avere una classe di dieci studenti e di classificarli in base ai risultati di cinque interrogazioni. Un tipico prospetto dei risultati per tale classe potrebbe essere

	Numero dello studente									
	1	2	3	4	5	6	7	8	9	10
1	92	71	81	52	75	97	100	63	41	75
2	85	63	79	49	71	91	93	58	52	71
Interrogazione 3	89	74	80	61	79	88	97	55	51	73
4	96	68	84	58	80	93	95	61	47	70
5	82	72	82	63	73	92	93	68	56	74

Useremo le istruzioni FOR...NEXT per leggere i dati contenuti nelle istruzioni DATA. Vogliamo che il computer calcoli e stampi le seguenti informazioni.

```
STUDENTE   MEDIA DEI VOTI
1          (Il computer stampa la media, ecc.)
2
3
(ecc.)
PROVA      MEDIA DELLA CLASSE
1          (Il computer stampa la media, ecc.)
2
3
(ecc.)
```

Sebbene le istruzioni DATA possano essere scritte in qualsiasi posto all'interno del programma, la prima istruzione dovrà essere un dimensionamento.

```
100 OPTION BASE 1
110 DIM G(5,10)
```

Così si predispone nella memoria lo spazio per una matrice con cinque righe e dieci colonne. Il numero di riga (R) sarà il numero di interrogazione mentre il numero di colonna (C) corrisponderà al numero di studente. Si possono inserire le istruzioni DATA.

```
120 DATA 92,71,81,52,75,97,9
9,63,41,75
130 DATA 85,63,79,49,71,91,9
3,58,52,71
140 DATA 89,84,80,61,79,88,9
7,55,51,73
150 DATA 96,68,84,58,80,93,9
5,61,47,70
160 DATA 82,72,82,63,73,92,9
3,68,56,74
```

Ora dobbiamo far leggere i dati al programma.

```
170 FOR R=1 TO 5
180 FOR C=1 TO 10
190 READ G(R,C)
200 NEXT C
210 NEXT R
```

In questo modo abbiamo introdotto riga per riga i numeri dentro la matrice G. Così i dati della linea 120 diventano la riga 1 della matrice G, i dati della linea 130 diventano la riga 2 della matrice, e così via. Prima di fare il resto stampiamo le intestazioni.

```
220 PRINT "STUDENTE","MEDIA
DEI VOTI"
230 PRINT
```

Adesso possiamo calcolare la media dei voti di ogni studente.

```
240 FOR C=1 TO 10
```

La linea 240 apre un ciclo che scorrerà attraverso le colonne della matrice. Per ogni valore di C calcoliamo la media di quella colonna e la stampiamo.

```
250 LET S=0
260 FOR R=1 TO 5
270 LET S=S+G(R,C)
280 NEXT R
290 PRINT C,S/5
```

Quindi il ciclo con variabile C viene chiuso.

```
300 NEXT C
```

Poi si ripete il procedimento per calcolare le medie sulle righe invece che sulle colonne.

```
310 PRINT
320 PRINT "PROVA","MEDIA DEL
LA CLASSE"
330 PRINT
340 FOR R=1 TO 5
350 LET S=0
```

```
360 FOR C=1 TO 10
370 LET S=S+G(R,C)
380 NEXT C
390 PRINT R,S/10
400 NEXT R
```

Infine c'è l'istruzione END.

```
410 END
```

Ecco il programma completo:

```
100 OPTION BASE 1
110 DIM G(5,10)
120 DATA 92,71,81,52,75,97,9
9,63,41,75
130 DATA 85,63,79,49,71,91,9
3,58,52,71
140 DATA 89,84,80,61,79,88,9
7,55,51,73
150 DATA 96,68,84,58,80,93,9
5,61,47,70
160 DATA 82,72,82,63,73,92,9
3,68,56,74
170 FOR R=1 TO 5
180 FOR C=1 TO 10
190 READ G(R,C)
200 NEXT C
210 NEXT R
220 PRINT "STUDENTE","MEDIA
DEI VOTI"
230 PRINT
240 FOR C=1 TO 10
250 LET S=0
260 FOR R=1 TO 5
270 LET S=S+G(R,C)
280 NEXT R
290 PRINT C,S/5
300 NEXT C
310 PRINT
320 PRINT "PROVA","MEDIA DEL
LA CLASSE"
330 PRINT
340 FOR R=1 TO 5
350 LET S=0
360 FOR C=1 TO 10
370 LET S=S+G(R,C)
380 NEXT C
390 PRINT R,S/10
400 NEXT R
410 END
```

Esempio 3 - Ordine alfabetico

Come esempio di un modo di usare un vettore stringa, costruiamo un programma che accetti in input una lista di parole, le metta in ordine alfabetico e stampi la lista ordinata.
Prima conveniamo di non mettere nella lista più di venti parole. Potrebbero essere in numero qualsiasi, ma per noi venti sono un numero abbastanza grande. Se chiamiamo A$ il vettore stringa, possiamo scrivere così le istruzioni per il dimensionamento.

```
100 OPTION BASE 1
110 DIM A$(20)
```

Poi, chiediamo quante parole ci sono nella lista in questione. Per quanto abbiamo stabilito, questo numero non potrà superare venti. Quindi inseriamo le parole.

```
120 PRINT "QUANTE SONO LE PA
ROLE";
130 INPUT N
140 FOR I=1 TO N
150 INPUT A$(I)
160 NEXT I
```

Ora che abbiamo la lista di parole, possiamo ordinarla. La parte del programma che segue assolve a questo compito.

```
170 FOR I=1 TO N-1
180 IF A$(I+1)>=A$(I)THEN 23
0
190 LET B$=A$(I+1)
200 LET A$(I+1)=A$(I)
210 LET A$(I)=B$
220 GOTO 170
230 NEXT I
```

Esaminate questo pezzo di programma finché non avrete capito come funziona. Se la condizione della linea 180 è vera, significa che le due parole che vengono confrontate sono già in ordine e si passa al prossimo confronto. Se è falsa, le istruzioni dalla linea 190 alla 210 le scambiano tra di loro. Poi, con la riga 220, i confronti ricominciano daccapo. Il procedimento continua finché la condizione della riga 170 è vera per tutta la lista di parole che, in tal caso, è in ordine alfabetico.
Non resta che stamparla.

```
240 PRINT
250 FOR I=1 TO N
260 PRINT A$(I)
270 NEXT I
280 END
```

Il programma completo è

```
100 OPTION BASE 1
110 DIM A$(20)
120 PRINT "QUANTE SONO LE PA
ROLE";
130 INPUT N
140 FOR I=1 TO N
150 INPUT A$(I)
160 NEXT I
170 FOR I=1 TO N-1
180 IF A$(I+1)>=A$(I)THEN 23
0
190 LET B$=A$(I+1)
200 LET A$(I+1)=A$(I)
210 LET A$(I)=B$
220 GOTO 170
```

```
230 NEXT I
240 PRINT
250 FOR I=1 TO N
260 PRINT A$(I)
270 NEXT I
280 END
```

Provate questo programma con delle parole scelte da voi. Verificate che si ottiene effettivamente una lista delle parole in ordine alfabetico.

Esempio 4 - Rubrica telefonica

Come esempio finale supponiamo che a una piccola ditta serva un elenco telefonico ordinato rispetto il numero di identificazione dei clienti. Le informazioni verranno date al computer con una matrice bidimensionale A$. I dati di ogni cliente occuperanno una riga come segue: colonna 1 – numero di identificazione del cliente, colonna 2 – cognome, colonna 3 – nome, colonna 4 – prefisso telefonico e colonna 5 – numero telefonico. L'elemento A$(0,0) conterrà N (il numero dei clienti della lista). Tutte le informazioni saranno variabili di stringa. Così, i valori numerici inseriti verranno trasformati in stringhe e verranno nuovamente convertiti in numeri quando verranno letti dalla memoria.
Per prima cosa dobbiamo dimensionare la matrice per un numero massimo di clienti. Poiché questo esempio è semplicemente dimostrativo, limiteremo il numero massimo di clienti a venti. Naturalmente in una situazione reale, questo numero sarebbe molto più grande. In ogni caso il problema consiste nello scrivere un programma per l'inserimento dei dati relativi ad N clienti, per collocare le informazioni in una matrice A$ e per trasferirle successivamente su un nastro magnetico. Si può iniziare il programma con facilità.

```
100 OPTION BASE 0
110 DIM A$(21,6)
```

Poi chiediamo il numero dei clienti che verranno inseriti.

```
120 PRINT "QUANTI SONO I NOM
I";
130 INPUT N
140 LET A$(0,0)=STR$(N)
```

Nel programma possiamo usare N, ma per inserire questo numero nella matrice dobbiamo prima trasformarlo in una stringa.
L'input dei dati e la loro collocazione nella matrice segue senza difficoltà.

```
150 FOR I=1 TO N
160 LET J=1
170 INPUT "CODICE CLIENTE= "
:B$
180 LET A$(I,J)=B$
190 LET J=J+1
200 INPUT "COGNOME= ":B$
210 LET A$(I,J)=B$
220 LET J=J+1
230 INPUT "NOME= ":B$
240 LET A$(I,J)=B$
250 LET J=J+1
260 INPUT "PREFISSO= ":B$
270 LET A$(I,J)=B$
280 LET J=J+1
290 INPUT "NUMERO TEL.= ":B$
300 LET A$(I,J)=B$
310 NEXT I
```

Ora che le informazioni sono state inserite, possiamo memorizzarle sul nastro magnetico.

```
320 OPEN #1:"CS1",OUTPUT,FIX
ED 64
330 PRINT #1:A$(0,0)
340 FOR R=1 TO N
350 FOR C=1 TO 5
360 PRINT #1:A$(R,C)
370 NEXT C
380 NEXT R
390 CLOSE #1
400 END
```

Segue il programma completo.

```
100 OPTION BASE 0
110 DIM A$(21,6)
120 PRINT "QUANTI SONO I NOM
I";
130 INPUT N
140 LET A$(0,0)=STR$(N)
150 FOR I=1 TO N
160 LET J=1
170 INPUT "CODICE CLIENTE= "
:B$
180 LET A$(I,J)=B$
190 LET J=J+1
200 INPUT "COGNOME= ":B$
210 LET A$(I,J)=B$
220 LET J=J+1
230 INPUT "NOME= ":B$
240 LET A$(I,J)=B$
250 LET J=J+1
260 INPUT "PREFISSO= ":B$
270 LET A$(I,J)=B$
280 LET J=J+1
290 INPUT "NUMERO TEL.= ":B$
300 LET A$(I,J)=B$
310 NEXT I
320 OPEN #1:"CS1",OUTPUT,FIX
ED 64
330 PRINT #1:A$(0,0)
340 FOR R=1 TO N
350 FOR C=1 TO 5
360 PRINT #1:A$(R,C)
370 NEXT C
380 NEXT R
390 CLOSE #1
400 END
```

Potete utilizzare questo programma con nomi e numeri a vostra scelta. Dopo che i dati sono stati memorizzati, vorremmo richiamare gli elementi della matrice A$ dal registratore. Il seguente programma serve a questo scopo.

```
100 OPTION BASE 0
110 DIM A$(21,6)
120 OPEN #1:"CS1",INPUT ,FIX
ED 64
130 INPUT #1:M$
140 LET A$(0,0)=M$
150 LET N=VAL(M$)
160 FOR R=1 TO N
170 FOR C=1 TO 5
180 INPUT #1:B$
190 LET A$(R,C)=B$
200 NEXT C
210 NEXT R
220 CLOSE #1
230 END
```

Naturalmente, quando la matrice A$ è stata riportata nella memoria del computer, la si può modificare, ordinare, se si desidera, e trasferire nuovamente sul nastro magnetico. Tuttavia lo scopo di questo esempio è di illustrare come una matrice può essere conservata su nastro per poi essere nuovamente richiamata da tale dispositivo.

PROBLEMI

1. Scrivete un programma che usi le istruzioni

```
200 DATA 12
210 DATA 2,1,4,3,2,4,5,6,3,5
,4,1
```

Esso leggerà le dimensioni di un vettore numerico dalla prima delle istruzioni DATA e dalla seconda leggerà gli elementi del vettore. Al termine fate stampare il vettore.

2. Scrivete un programma Basic che legga venticinque numeri da istruzioni DATA e li metta in un vettore A. Fate in modo che cerchi e stampi tutti gli elementi il cui valore è maggiore di cinque. Potete scegliere a piacimento i numeri da inserire nelle istruzioni DATA.

3. Cosa si otterrà dall'esecuzione del seguente programma?

```
100 OPTION BASE 1
110 DIM Y(6)
120 FOR I=1 TO 6
130 READ Y(I)
140 NEXT I
```

```
150 DATA 2,1,3,1,2,1
160 LET S1=0
170 LET S2=0
180 FOR I=1 TO 6
190 LET S1=S1+Y(I)
200 LET S2=S2+Y(I)^2
210 NEXT I
220 X=S2-S1
230 PRINT X
240 END
```

4. Quale sarà il risultato del seguente programma?

```
100 OPTION BASE 1
110 DIM A(10)
120 FOR I=1 TO 10
130 READ A(I)
140 NEXT I
150 LET X=A(1)
160 FOR I=1 TO 9
170 LET A(I)=A(I+1)
180 NEXT I
190 LET A(10)=X
200 FOR I=1 TO 10
210 PRINT A(I)
220 NEXT I
230 DATA 10,9,8,7,6,5,4,3,2,
1
240 END
```

5. Scrivete un programma Basic per ricevere in input N (un numero intero tra 1 e 100) e poi un vettore di N elementi. Fate ordinare il vettore in ordine decrescente per poi stamparlo. (Suggerimento: guardate il programma esempio numero 3.)

6. Supponiamo che il primo numero nelle istruzioni DATA esprima il numero dei dati che lo seguono e che questi siano tutti numeri interi tra 1 e 10, estremi compresi. Scrivete un programma che conti quanti 1, quanti 2, ecc. sono stati inseriti tra i dati e poi stampi i risultati. (Suggerimento: cercate di catalogare i dati mentre vengono letti e di incrementare di volta in volta un elemento di un vettore che raccolga i risultati.)

7. Che stampa si otterrà eseguendo il seguente programma?

```
100 OPTION BASE 1
110 DIM Z(6,6)
120 FOR R=1 TO 6
130 FOR C=1 TO 6
140 LET Z(R,C)=0
150 NEXT C
160 NEXT R
170 FOR R=1 TO 5 STEP 2
180 FOR C=R TO 6
190 LET Z(R,C)=1
200 NEXT C
210 NEXT R
220 FOR R=1 TO 6
230 FOR C=1 TO 6
240 PRINT Z(R,C);
250 NEXT C
```

```
260 PRINT
270 PRINT
280 NEXT R
290 END
```

8. Se si facesse funzionare il programma che segue, cosa si otterrebbe?

```
100 OPTION BASE 1
110 DIM A(5,5)
120 FOR R=1 TO 5
130 FOR C=1 TO 5
140 LET A(R,C)=2
150 NEXT C
160 NEXT R
170 FOR C=5 TO 1 STEP -1
180 FOR R=1 TO C
190 LET A(R,C)=3
200 NEXT R
210 NEXT C
220 FOR R=1 TO 5
230 FOR C=1 TO 5
240 PRINT A(R,C);
250 NEXT C
260 PRINT
270 PRINT
280 NEXT R
290 END
```

9. Scrivete un programma che legga la seguente matrice con istruzioni DATA e poi la stampi.

$$\begin{bmatrix} 2 & 1 & 0 & 5 & 1 \\ 3 & 2 & 1 & 3 & 1 \end{bmatrix}$$

10. Costruite un programma che legga con istruzioni DATA la seguente matrice e poi la stampi.

$$\begin{bmatrix} 5 & 3 \\ 2 & 0 \\ -1 & 1 \\ 4 & 2 \\ 2 & 6 \end{bmatrix}$$

11. Scrivete un programma Basic che riceva in input una matrice M per N. Fategli calcolare e stampare la somma degli elementi di ogni riga e il prodotto degli elementi di ogni colonna.

12. Scrivete un programma Basic che legga due matrici da istruzioni DATA. Siano entrambe di dimensioni due per tre. Costruite poi un'altra matrice due per tre tale che ogni suo elemento sia la somma dei corrispondenti elementi delle altre due matrici. Fate stampare la terza matrice.

13. I dati elencati qui sotto sono le vendite realizzate da dei commessi viaggiatori in una settimana.

		Lun	Mar	Mer	Gio	Ven	Sab
Venditore	1	48	40	73	120	100	90
	2	75	130	90	140	110	85
	3	50	72	140	125	106	92
	4	108	75	92	152	91	87

Scrivete un programma che calcoli e stampi (a) le vendite totali giornaliere, (b) le vendite totali settimanali di ogni commesso viaggiatore e (c) le vendite totali della settimana.

14. Scrivete un programma Basic che raccolga in due differenti vettori N nomi ed N voti. Supponiamo che N non sia maggiore di venti. Trattate il vettore dei nomi in modo che questi risultino in ordine alfabetico ed il vettore dei voti in modo che i voti siano accoppiati correttamente alle persone. Provate il programma con dati a vostra scelta.

15. Con gli stessi dati del problema 14, ordinate i voti in maniera decrescente e fate in modo che le persone risultino abbinate esattamente ai loro voti.

16. Scrivete un programma per memorizzare sul nastro magnetico dieci numeri introdotti da tastiera.

17. Scrivete un programma che accetti in input, da nastro magnetico, dieci nomi propri. Fate poi in modo che la lista risulti ordinata in ordine alfabetico e venga stampata.

ESERCIZI

Provate a risolvere i seguenti esercizi. Troverete le risposte in fondo al libro.

1. A cosa servono le istruzioni DIM e OPTION?

---

2. Consideriamo una matrice X. In Basic quale nome di variabile si usa per individuare l'elemento della riga 3, colonna 4?

---

3. Cosa succederà se si esegue il seguente programma?

```
100 OPTION BASE 1
110 DIM A$(4),B(4)
120 FOR I=1 TO 4
130 READ A$(I),B(I)
140 NEXT I
150 PRINT A$(4),B(2)
160 DATA "LUCA",165,"PAOLO",
183
170 DATA "MARCO",145,"CARLO"
,192
180 END
```

---

4. Scrivete un programma che riceva in input una lista di numeri e che stampi la somma dei numeri positivi.

---

5. Supponiamo di avere una matrice di stringhe X. Quale nome di variabile viene usato in Basic per individuare l'elemento della riga 2, colonna 4?

---

6. Scrivete un programma che usi le istruzioni FOR...NEXT per riempire di quattro (4) una matrice quattro per sei. Fatela poi stampare.

---

7. Quale sarà l'output del seguente programma?

```
100 OPTION BASE 1
110 DIM A(5,5)
120 FOR I=1 TO 5
130 FOR J=1 TO 5
140 LET A(I,J)=0
150 NEXT J
160 NEXT I
170 FOR I=1 TO 5
180 LET A(I,I)=2
190 NEXT I
200 FOR I=1 TO 5
210 FOR J=1 TO 5
220 PRINT A(I,J);
230 NEXT J
240 PRINT
250 PRINT
260 NEXT I
270 END
```

---

8. Sia questa la matrice A:

$$\begin{bmatrix} 1 & 3 & 5 \\ 6 & 2 & 4 \end{bmatrix}$$

a. Scrivete una istruzione DIM per A.

____

b. Qual è il valore di A(2,3)?

____

c. Se X = 1 e Y = 2, cos'è A(X,Y)?

____

d. Quant'è A(A(1,1),A(2,2))?

____

9. A cosa serve l'istruzione OPEN?

____

10. Qual è lo scopo dell'istruzione CLOSE?

____

CAPITOLO NOVE

Funzioni definite dall'utente e subroutine

ARGOMENTI

In questo capitolo studieremo come si può programmare il computer per compiere delle sottoperazioni. Ciò si può fare con parti di programma o con speciali istruzioni "in linea". In particolare ecco i nostri argomenti:

Le funzioni definite dall'utente

Abbiamo già incontrato precedentemente le funzioni interne del Basic. Impareremo ora a definire da noi stessi delle funzioni per scopi specifici.

Le subroutine

Le subroutine sono molto utili quando si devono ripetere più volte delle operazioni complicate. Vedremo come esse si costruiscono e come si usano nei programmi Basic.

Applicazioni

Spesso il principiante non riesce a capire il valore delle funzioni definite dall'utente e delle subroutine. Sottoporremo ripetutamente alla vostra attenzione questi concetti della programmazione in Basic.

PRATICA SUL CALCOLATORE

---

1. Accendete il computer e scrivete il seguente programma:

```
100 DEF FNA(X)=5*X+4
110 LET X=2
120 LET Y=5*X+4
130 PRINT Y,FNA(2)
140 END
```

Eseguitelo e riportate sotto il risultato.

____

2. Cambiate la riga 130 con

```
130 PRINT Y,FNA(X)
```

Fate una lista del programma. Cosa pensate che si otterrà in esecuzione?

____

3. Sostituite la riga 110 con questa

```
110 LET X=5
```

Listate il programma ed esaminatelo attentamente. Quale sarà il suo output, se lo eseguiamo?

____

Controllate l'esattezza della vostra risposta. Eseguitelo e segnate ciò che è accaduto.

____

4. Modificate ora la riga 130 così:

```
130 PRINT Y,FNA(5)
```

Fate una lista. Cosa farà, secondo voi, il programma?

Fatelo funzionare e prendete nota dei risultati.

---

5. Notate che le espressioni dopo i segni di uguale delle righe 100 e 120 del programma sono la stessa. In una delle versioni del programma abbiamo stampato Y e FNA(X) ed abbiamo visto che erano uguali. Continuiamo su questa strada. Cancellate il programma in memoria ed introducete il seguente:

```
100 DEF FNA(X)=X^2
110 DEF FNB(X)=3*X
120 DEF FNC(X)=X+2
130 LET X=1
140 PRINT FNA(X),FNB(X),FNC(X)
150 END
```

Studiatelo attentamente. Cosa pensate che verrà stampato se lo si esegue?

Fate funzionare il programma e scrivete cosa è successo.

Sostituite 1 al posto di X nelle espressioni a destra dell'uguale nelle righe 100, 110 e 120 del programma. Trascrivete i numeri che ottenete.

Confrontateli con quelli stampati dal computer.

6. Modificate la riga 130 in modo da avere

```
130 LET X=2
```

Fate una lista del programma. Che risultato darà ora l'esecuzione?

Controllate se avete risposto correttamente. Eseguite il programma e annotate sotto ciò che avete ottenuto.

---

7. Bene, cambiate la riga 130 con

```
130 LET X=3
```

Cosa succederà facendo funzionare il programma?

Verificate l'esattezza della vostra risposta eseguendo il programma e annotando ciò che è successo.

---

8. Mettiamo in luce con lo stesso programma alcuni altri concetti. Scrivete

```
130 LET X=1
140 PRINT FNC(X+4),FNA(X),FN
B(2)
```

e fate una lista. Scrivete che cosa pensate che stamperà il programma se viene eseguito.

Fatelo funzionare e prendete nota del risultato.

---

9. Tentiamo una piccola variazione sullo stesso tema. Scrivete

```
140 PRINT FNA(X),FNB(FNA(X))
```

Listate il programma e analizzatelo con cura. Pensate a cosa verrà stampato in esecuzione. Segnate sotto la vostra risposta.

Eseguite il programma e controllate se avete sbagliato. Riportate sotto ciò che è accaduto.

---

10. Ancora una cosa su questo argomento. Scrivete

```
130 LET X=4
140 PRINT FNA(X),FNC(X),FNA(
SQR(X))
```

Cosa accadrà ora nel programma?

____

Eseguitelo e prendete nota di ciò che è successo.

____

11. Fino ad ora abbiamo usato soltanto numeri nelle istruzioni DEF. Dobbiamo anche usare istruzioni DEF che operino su stringhe. Cancellate la memoria e scrivete il seguente programma:

```
100 DEF SPAZIO$(A$)=SEG$(A$,
1,2)&CHR$(32)&SEG$(A$,3,LEN(
A$)-2)
110 INPUT N$
120 PRINT SPAZIO$(N$)
130 GOTO 110
140 END
```

La funzione definita alla linea 100 si chiama SPAZIO$. Il simbolo $ alla fine del nome indica che la funzione opera su stringhe. Studiate brevemente la definizione di SPAZIO$. Eseguite il programma e, al segnale di input, scrivete CARLO. Cosa appare?

____

Ora scrivete SARA. Cosa è successo?

____

A questo punto vi sarete accorti che SPAZIO$ inserisce uno spazio tra il primo e il secondo carattere della stringa sulla quale opera la funzione. Non badiamo all'utilità o meno di questa funzione. Il nostro scopo è quello di dimostrare in che modo le istruzioni DEF possono agire sulle stringhe. Fate uscire il computer dal ciclo di input.

12. Cancellate il programma dalla memoria e inserite il seguente nuovo programma.

```
100 DEF PI=3.141592654
110 INPUT "RAGGIO = ":RAGGIO
120 LET CIRC=2*PI*RAGGIO
130 PRINT "CIRCONFERENZA =";
```

```
CIRC
140 GOTO 110
150 END
```

Questo programma è semplice e il suo scopo è evidente. La linea 100 illustra ancora un altro tipo di istruzione DEF. Eseguite il programma e provatelo con diversi valori numerici. Al termine fate uscire il computer dal ciclo di input.

13. Cancellate il programma in memoria e scrivete questo:

```
100 PRINT "A";
110 GOSUB 200
120 PRINT "B";
130 GOSUB 300
140 PRINT "C";
150 STOP
200 PRINT 1;
210 RETURN
300 PRINT 2;
310 RETURN
400 END
```

Questo programma contiene tre istruzioni nuove: GOSUB, RETURN e STOP ed è stato costruito solo per mostrare a che cosa servono. Fatelo funzionare e prendete nota del risultato.

Mettete in relazione ciò che è stato stampato con le righe di programma corrispondenti.

14. A quale istruzione passa il controllo il GOSUB della riga 110? (Suggerimento: guardate la stampa del passo 13.)

---

15. A quale istruzione fa tornare il programma il RETURN della riga 210? (Suggerimento: di nuovo, esamiante la stampa nel passo 13.)

---

16. Le etichette sotto elencate danno l'ordine in cui il programma esegue le varie operazioni.

Etichetta	Operazione
100	Stampa A
110	Passa alla riga 200
200	Stampa 1
210	Passa alla riga 120
120	Stampa B
130	Passa alla riga 300
300	Stampa 2
310	Passa alla riga 140
140	Stampa C
150	Passa alla riga 400
400	Fine del programma

Leggete la sequenza attentamente e seguite l'evolversi del programma sulla lista. Riuscite già a capire qual'è la funzione delle istruzioni GOSUB e RETURN? E quella dello STOP?

---

17. Cancellate il programma dalla vostra area di lavoro. Scrivete invece quest'altro:

```
100 REM SUBR. DEMO
110 DIM X(4)
120 READ X(1),X(2),X(3),X(4)
130 REM SORT
140 GOSUB 300
150 REM PRINT
160 GOSUB 400
170 LET X(3)=7
180 REM SORT
190 GOSUB 300
200 REM PRINT
210 GOSUB 400
220 STOP
300 REM SORT SUBR.
310 FOR I=1 TO 3
320 IF X(I+1)>X(I)THEN 370
330 LET C=X(I+1)
340 LET X(I+1)=X(I)
350 LET X(I)=C
360 GOTO 310
370 NEXT I
380 RETURN
400 REM PRINT SUBR.
410 PRINT X(1),X(2),X(3),X(4
)
420 RETURN
500 DATA 2,1,5,6
600 END
```

Fate una lista ed assicuratevi di averlo scritto giusto. Questo programma ci dà un esempio di come si può adoperare una subroutine. La subroutine che va dalla riga 300 alla 380 pone il vettore X in ordine crescente. La subroutine delle righe 400, 410 e 420 stampa il vettore. Eseguite il programma e prendete nota del risultato.

Si noti che il vettore iniziale è

2 1 5 6

Ciò si può vedere analizzando l'istruzione DATA del programma. Nella riga 140 il programma salta alla subroutine che ordina gli elementi. Quando il controllo ritorna alla riga 150, il vettore è nella forma

1 2 5 6

Nella riga 170 sostituiamo il terzo elemento del vettore e poi passiamo di nuovo alla subroutine per eseguire l'ordinamento. Dopo esser tornato alla riga 200, il programma stampa il vettore

1 2 6 7

Infine il comando STOP della riga 220 fa sì che il programma salti all'istruzione END. Chiaramente possiamo usare l'istruzione GOSUB 300 per ordinare il vettore ogniqualvolta lo vogliamo. Questo è certamente più comodo che riscrivere le operazioni di ordinamento ogni volta.

18. Anche per questo capitolo la parte di pratica al calcolatore è finita.

DISCUSSIONE

Vogliamo studiare più approfonditamente i concetti introdotti nel paragrafo precedente. Quando avrete capito bene come il computer si serve di essi, allora sarete in grado di realizzare programmi molto più validi di quelli che riuscite a fare ora.

Le funzioni definite dall'utente

L'istruzione DEF (abbreviazione "definiamo") ci consente di aggiungere delle funzioni a quelle interne del linguaggio Basic (SQR, INT ecc.). Le istruzioni DEF possono essere sia numeriche che di stringa. Il modo più semplice per imparare questa funzione è di considerare alcuni tipici esempi.

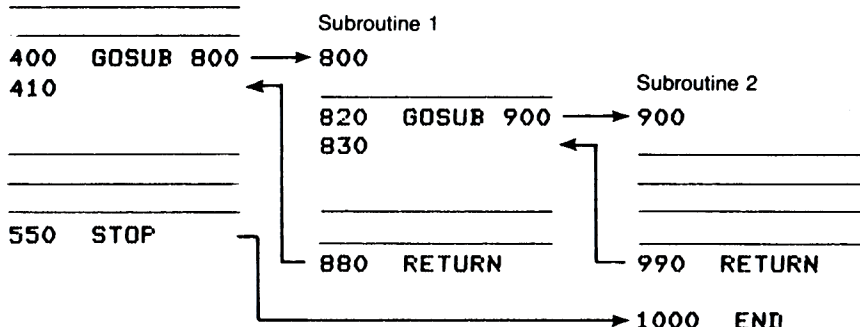
```
100 DEF FNA(X)=X*4-1
110 DEF PI=3.141592654
120 DEF TASSO(N)=(N-20)*.15
130 DEF SCAMBIO$(S$)=SEG$(S$
,2,LEN(S$)-1)&SEG$(S$,1,1)
```

Discutendo su come funziona ciascuno di questi semplici esempi possiamo vedere immediatamente come le istruzioni DEF possono essere usate vantaggiosamente nei programmi. L'istruzione DEF nella linea 100 è di immediata comprensione. Se in un programma si usa la variabile FNA(2), il computer capirebbe di dover sostituire alla X il valore 2 nell'espressione a destra dell'istruzione DEF. Il risultato è che FNA(2) sarebbe calcolata uguale a sette. Analogamente, se Y è uguale a sei, FNA(Y) assumerebbe il valore di ventitre. Addirittura potremmo scrivere in questo modo: FNA(SQR(Z) + 1.5). Il punto è che l'argomento della funzione (ciò che appare tra parentesi dopo FNA) viene convertito in un numero il quale è poi sostituito alla X nell'istruzione DEF.
La funzione DEF nella linea 110 è molto utile. Nei programmi si usano spesso delle costanti. In questo esempio, la costante PI è definita uguale a 3.141592654. In seguito, nel programma, potremo usare PI invece del corrispondente valore numerico. Questa possibilità è particolarmente utile quando le costanti sono comunemente ricordate con i loro nomi piuttosto che con i loro valori numerici. Naturalmente, se preferiamo, possiamo specificare le costanti anche con l'istruzione LET.
Lo scopo dell'istruzione DEF nella linea 120 è quello di dimostrare che in questo tipo di istruzioni possiamo usare tutti i nomi che desideriamo. In questo esempio, il tasso è il 15% della differenza tra N e venti. L'istruzione DEF associa questo valore alla funzione TAX(N).
L'ultimo esempio si trova alla linea 130. Qui l'istruzione DEF riguarda stringhe. La funzione SCAMBIO$ sposta un carattere dall'inizio di una stringa alla fine. Ad esempio CASA diventa ASAC, UCCELLO diventa CCELLOU, e così via.
Il vantaggio principale di queste funzioni definite dall'utente con l'istruzione DEF è quello di semplificare la programmazione evitando di ripetere l'uso di espressioni complicate. Le istruzioni DEF implementate nel TI home computer sono tra le più potenti che si possono trovare nella maggior parte delle versioni di Basic. Il programmatore esperto coglie al volo le opportunità di risparmiare fatica usando le istruzioni DEF.

Le subroutine

Uno dei limiti delle istruzioni DEF è che possono contenere una sola variabile e che l'espressione deve stare tutta su di una riga.
Ma possono capitare situazioni in cui bisogna ripetere più volte operazioni molto più complesse. Ecco perché ci sono le subroutine. Lo schema sotto mostra come si può usare una subroutine.

Inizio del programma principale	______________
	______________
	______________
	200 GOSUB 1000
	210
	______________
	______________
	______________
	350 GOSUB 1000
	360
	______________
	______________
Fine del programma principale	430 STOP
Inizio della subroutine	1000 REM SUBROUTINE
	______________
	______________
	______________
Fine della subroutine	1150 RETURN
Fine del programma	1200 END

In esecuzione, quando il computer raggiunge il GOSUB della riga 200, il programma passa all'inizio della subroutine nella riga 1000. Dopo che la subroutine è stata eseguita, il RETURN che si trova nella riga 1150 passa il controllo alla riga seguente a quella dove c'è il GOSUB che richiamava la subroutine. In questo caso il programma salta alla riga 210. Il programma principale continua fino alla riga 350, che passa di nuovo il controllo alla subroutine della riga 1000. E questa volta il RETURN fa saltare il programma alla riga 360. Naturalmente possiamo usare l'istruzione GOSUB 1000 quante volte vogliamo e avere un gran numero di subroutine. Di solito la prima parte del programma è il programma principale mentre le subroutine sono tutte raggruppate alla fine. C'è una buona ragione per far questo. Noi vogliamo che le subroutine vengano eseguite solo quando sono richiamate da GOSUB. Così, quando termina il programma principale, inseriamo una istruzione STOP. Ciò è equivalente a un GOTO alla END e salta tutte le subroutine raggruppate alla fine del programma. Possiamo usare l'istruzione STOP in qualsiasi punto del programma dove ci sia una fine logica. Ciò può succedere anche più volte in un solo programma.
È possibile e talvolta molto utile, saltare da una subroutine all'altra. Lo schema sotto mostra come ciò può accadere.

Si noti che il controllo passa da 400 a 800, poi da 820 a 900 fino al RETURN della riga 990. Il problema qui è di sapere se il RETURN ci porta alla riga 410 o alla riga 830. La risposta è che il RETURN ci riporta all'istruzione seguente il GOSUB che ci ha mandato in quella subroutine. Poiché il GOSUB ci ha richiamato la subroutine 2 è alla riga 820, il RETURN della riga 990 ci rimanda alla 830. La stessa regola si applica quando si raggiunge il RETURN della riga 880. Siamo entrati nella subroutine 1 a causa del GOSUB della riga 400. Così il RETURN con etichetta 880 ci riporta alla riga 410. Infine, la istruzione STOP della riga 550 passa il controllo all'END con etichetta 1000.
Forse ora vi chiederete perché le subroutine sono così importanti. Ma ciò vi risulterà chiaro quando avrete acquisito più esperienza come programmatori. Per adesso vi basti sapere che la loro importanza è molto grande: sono uno dei mezzi più potenti a disposizione di chi programma.

ESEMPI DI PROGRAMMI

Ci sono parecchi esempi che aiutano ad approfondire i concetti di funzione definita dall'utente e di subroutine.

Esempio 1 - Arrotondamento dei valori, in dollari, ai centesimi

Alcune applicazioni di tipo commerciale richiedono di stampare dei risultati fino ai centesimi. Poiché il computer, nei calcoli, considera dieci cifre significative, si potrebbe avere la stampa di un numero come 23.15976431. Ma questo risultato è scomodo e ci piacerebbe ottenere la cifra arrotondata ai centesimi, cioè 23.16.

Quel che ci serve è proprio una funzione definita dall'utente. Scriviamo un programma che dia in esecuzione questo output:

```
PREZZO DELL'ARTICOLO ? 22.80
COL 10% DI SCONTO E' 20.52
COL 15% DI SCONTO E' 19.38
COL 20% DI SCONTO E' 18.24
```

Tutti i valori da stampare devono essere arrotondati al centesimo. Come prima cosa, dobbiamo definire una funzione che effettui l'arrotondamento. Eccola:

```
100 DEF FNR(X)=INT(X*100+.5)
/100
```

Per vedere come questa regola funziona, consideriamo adesso X 23.15976431. Vediamo cosa succede al valore X passo per passo.

X*100 = 2315.976431
X*100+0.5 = 2316.476431
INT(X*100+0,5) = 2316
INT(X*100+0.5)/100 = 23.16

Dunque 23.15976431 viene arrotondato esattamente a 23.16. Come secondo esempio, sia X = 23.15472563. Allora

X*100 = 2315.472563
X*100+0.5 = 2315.972563
INT(X*100+0.5) = 2315
INT(X*100+0.5)/100 = 23.16

Che significa che il valore arrotondato di 23.15472563 è 23.15. Le seguenti righe di programma non hanno bisogno di commenti

```
110 PRINT "PREZZO DELL'ARTIC
OLO";
120 INPUT Z
130 PRINT "COL 10% DI SCONTO
 E'";FNR(.9*Z)
140 PRINT "COL 15% DI SCONTO
 E'";FNR(.85*Z)
150 PRINT "COL 20% DI SCONTO
 E'";FNR(.8*Z)
```

Volendo possiamo tornare all'inizio con

```
160 GOTO 110
```

e quindi chiudere il programma con END

```
170 END
```

Ecco la lista completa:

```
100 DEF FNR(X)=INT(X*100+.5)
/100
110 PRINT "PREZZO DELL'ARTIC
OLO";
120 INPUT Z
130 PRINT "COL 10% DI SCONTO
 E'";FNR(.9*Z)
140 PRINT "COL 15% DI SCONTO
 E'";FNR(.85*Z)
150 PRINT "COL 20% DI SCONTO
 E'";FNR(.8*Z)
160 GOTO 110
170 END
```

Abbiamo usato la funzione da noi definita nelle righe 130, 140 e 150. Col 10 per cento di sconto il prezzo di vendita dell'articolo è il 90 per cento del prezzo originale Z. Quindi quello che stampiamo è FNR(09*Z), cioè il valore arrotondato ai centesimi. Si noti che l'uso della funzione che abbiamo definito ci evita di dover scrivere ogni volta l'espressione della riga 100.

Esempio 2 - Determinare il prezzo di una moquette (in dollari)

Vogliamo scrivere un programma che si serva di una subroutine per calcolare il prezzo di istallazione di una moquette. Supponiamo che ci siano quattro tipi di moquette e che il prezzo di ciascuno sia scontato in base alla quantità acquistata. Sia questo il listino dei prezzi:

		Prezzo per metro quadro		
		1	2	3
Tipo	A	$10.00	$ 8.50	$ 7.25
	B	13.25	12.00	9.75
	C	16.00	14.00	11.25
	D	20.00	17.20	15.25

1: Per i primi 15 metri quadri
2: Per un qualsiasi dimensione tra i 15 e i 25 metri quadri
3: Oltre i 25 metri quadri.

L'output del programma dovrebbe essere il seguente

```
QUANTE SONO LE STANZE ? 4
PER OGNI STANZA SCRIVERE
LA LUNGHEZZA E LA LARGHEZZA
SEPARATE DA UNA VIRGOLOLA,
ESPRESSE IN METRI

STANZA      DIMENSIONI

1           ? 3,4
2           ? 4,5
3           ? 4,2.5
4           ? 6,8
90.00 METRI QUADRI RICHIESTI

TIPO DI     PREZZO
MOQUETTE    TOTALE

A           652.50
B           877.50
C           1012.50
D           1372.50
```

Prima di addentrarci nel programma, riflettiamo un po' sull'output. Siccome il risultato è in dollari e cent, possiamo usare la funzione definita dall'utente dell'esempio 1 per arrotondare esattamente le cifre. Possiamo servircene anche per arrotondare alle centinaia il numero di metri di moquette richiesti. Cominciamo dunque il programma con questa funzione.

```
100 DEF FNR(X)=INT(X*100+.5)
/100
```

Le prime righe non presentano difficoltà.

```
110 PRINT "QUANTE SONO LE ST
ANZE";
120 INPUT N
130 PRINT "PER OGNI STANZA S
CRIVERE"
140 PRINT "LA LUNGHEZZA E LA
 LARGHEZZA IN METRI,"
150 PRINT "SEPARATE DA UNA V
IRGOLA"
160 PRINT
170 PRINT "STANZA","DIMENSIO
NI"
180 PRINT
```

Ora siamo pronti a inserire le dimensioni della stanza. Useremo la variabile AREA per calcolare l'area. Per farlo, si moltiplicano fra loro lunghezza e larghezza.

```
190 LET AREA=0
200 FOR I=1 TO N
210 PRINT I,
220 INPUT L,W
230 LET AREA=AREA+L*W
240 NEXT I
```

Adesso stampiamo la quantità di moquette richiesta, arrotondata ai centesimi.

```
250 PRINT FNR(AREA);"METRI Q
UADRI RICHIESTI"
```

A questo punto possiamo benissimo introdurre nel programma la tabella dei prezzi per mezzo dell'istruzione DATA.

```
260 DATA 10,8.5,7.25
270 DATA 13.25,12,9.75
280 DATA 16,14,11.25
290 DATA 20,17.2,15.25
```

Quindi stampiamo l'introduzione alla stampa dei prezzi.

```
300 PRINT
310 PRINT "TIPO DI","PREZZO"
320 PRINT "MOQUETTE","TOTALE
"
330 PRINT
```

Arriviamo ora al punto dove si userà la subroutine. Poiché non possiamo sapere di preciso dove comincerà la subroutine, le assegniamo un'etichetta molto alta e poi, se necessario, la correggeremo.

```
340 REM CALCOLA IL COSTO DEL
 TIPO A
350 GOSUB 800
```

E adesso scriviamo la subroutine. Come prima cosa ci servono i tre prezzi per ogni tipo di moquette. Li leggiamo con le istruzioni DATA.

```
800 REM SUBROUTINE CHE CALCO
LA IL COSTO DELLA MOQUETTE
810 READ C1,C2,C3
```

Poi andiamo a vedere se l'area di moquette è minore di 15 metri quadri, se è tra 15 e 25 o se supera 25 e ci calcoliamo il costo di conseguenza.

```
820 IF AREA>25 THEN 860
830 IF AREA>15 THEN 880
840 LET P=C1*AREA
850 GOTO 890
860 LET P=15*C1+10*C2+(AREA-
25)*C3
870 GOTO 890
880 LET P=15*C1+(AREA-15)*C2
890 RETURN
```

Analizzate queste istruzioni una ad una per convicervi che il calcolo è effettivamente giusto. Ora possiamo tornare al programma principale e stampare il primo costo.

```
360 PRINT "A",FNR(P)
```

Una volta trovato il procedimento, si scrive facilmente anche il resto del programma principale.

```
370 REM CALCOLA IL COSTO DEL
 TIPO B
380 GOSUB 800
390 PRINT "B",FNR(P)
400 REM CALCOLA IL COSTO DEL
 TIPO C
410 GOSUB 800
420 PRINT "C",FNR(P)
430 REM CALCOLA IL COSTO DEL
 TIPO D
440 GOSUB 800
450 PRINT "D",FNR(P)
460 STOP
```

L'istruzione **STOP** nella riga **460** serve per evitare che il programma prosegua eseguendo la subroutine. L'importanza della subroutine appare chiara se si pensa che, non avendola a disposizione, bisognerebbe sostituire ad una istruzione **GOSUB** tante istruzioni quante ce ne sono nella subroutine in questione.
Ecco il programma completo:

```
100 DEF FNR(X)=INT(X*100+.5)
/100
110 PRINT "QUANTE SONO LE ST
ANZE";
120 INPUT N
130 PRINT "PER OGNI STANZA S
CRIVERE"
140 PRINT "LA LUNGHEZZA E LA
 LARGHEZZA IN METRI,"
150 PRINT "SEPARATE DA UNA V
IRGOLA"
160 PRINT
170 PRINT "STANZA","DIMENSIO
NI"
180 PRINT
190 LET AREA=0
200 FOR I=1 TO N
210 PRINT I,
220 INPUT L,W
230 LET AREA=AREA+L*W
240 NEXT I
250 PRINT FNR(AREA);"METRI Q
UADRI RICHIESTI"
260 DATA 10,8.5,7.25
270 DATA 13.25,12,9.75
280 DATA 16,14,11.25
290 DATA 20,17.2,15.25
300 PRINT
310 PRINT "TIPO DI","PREZZO"
320 PRINT "MOQUETTE","TOTALE
"
330 PRINT
340 REM CALCOLA IL COSTO DEL
 TIPO A
350 GOSUB 800
360 PRINT "A",FNR(P)
370 REM CALCOLA IL COSTO DEL
 TIPO B
380 GOSUB 800
390 PRINT "B",FNR(P)
400 REM CALCOLA IL COSTO DEL
 TIPO C
410 GOSUB 800
420 PRINT "C",FNR(P)
```

```
430 REM CALCOLA IL COSTO DEL
 TIPO D
440 GOSUB 800
450 PRINT "D",FNR(P)
460 STOP
800 REM SUBROUTINE CHE CALCO
LA IL COSTO DELLA MOQUETTE
810 READ C1,C2,C3
820 IF AREA>25 THEN 860
830 IF AREA>15 THEN 880
840 LET P=C1*AREA
850 GOTO 890
860 LET P=15*C1+10*C2+(AREA-
25)*C3
870 GOTO 890
880 LET P=15*C1+(AREA-15)*C2
890 RETURN
900 END
```

Esempio 3 - Inventario domestico

Come esempio finale scriveremo un programma per elaborare informazioni relative ai beni domestici e per registrarle su un nastro magnetico. Le informazioni sono quelle che sarebbero necessarie per un indennizzo assicurativo nell'eventualità che la vostra casa venisse danneggiata dal fuoco.

Le informazioni verranno scritte sotto forma di record (un blocco di caratteri) lunghi 51 caratteri. Lo spazio non utilizzato in ogni singolo record verrà riempito con degli spazi vuoti. Il primo carattere sarà uno spazio. I caratteri da due a sedici conterranno il nome della stanza. I caratteri da 17 a 31 conterranno il nome dell'articolo. In entrambi questi campi se non verranno utilizzati tutti i 15 caratteri che li compongono bisognerà completarli con spazi vuoti.

I caratteri 32 e 33 conterranno l'anno d'acquisto dell'articolo. I caratteri da 34 a 42 conterranno il prezzo iniziale dell'articolo mentre il suo valore corrente verrà posto nei caratteri da 43 a 51. Anche in questo caso bisognerà riempire con spazi vuoti le posizioni non utilizzate.

Il programma dovrebbe chiedere in input le informazioni necessarie, controllare la loro correttezza, convertire tutte le quantità numeriche in stringhe, riunire i cinquantun caratteri in un record che sia la descrizione di un articolo ed infine scrivere tale record sul nastro magnetico. Poiché a questo punto avete già acquisito una notevole esperienza con il computer, non esamineremo, come le altre volte, l'esempio in dettaglio e vi daremo subito il programma completo.

Dovreste considerarlo attentamente per comprendere esattamente come lavora. Poiché illustra come si possono usare le subroutine questo esempio costituisce un buon ripasso degli argomenti precedentemente discussi nel libro.

```
100 OPEN #1:"CS1",OUTPUT,FIX
ED 51
110 LET A$=" "
115 INPUT "STANZA":X$
120 GOSUB 700
130 INPUT "ARTICOLO":X$
140 GOSUB 700
150 INPUT "ANNO D'ACQUISTO":
X$
160 LET X$=SEG$(X$,LEN(X$)-1
,2)
170 LET A$=A$&X$
180 INPUT "PREZZO D'ACQUISTO
":P
190 GOSUB 800
200 INPUT "VALORE CORRENTE":
P
210 GOSUB 800
220 PRINT #1:A$
230 GOTO 110
500 REM SUBR. PER COMPLETARE
 LA STRINGA CON SPAZI VUOTI
505 LET X$=" "
510 FOR I=1 TO N
520 LET X$=X$&CHR$(32)
530 NEXT I
540 RETURN
600 REM SUBR. PER ANTEPORRE
ALLA STRINGA SPAZI VUOTI
605 LET X$=" "
610 FOR I=1 TO N
620 LET X$=CHR$(32)&X$
630 NEXT I
640 RETURN
700 REM CONTROLLO LUNGHEZZA
STRINGA
710 IF LEN(X$)<=15 THEN 740
720 LET X$=SEG$(X$,1,15)
730 GOTO 760
740 LET N=15-LEN(X$)
750 GOSUB 500
760 LET A$=A$&X$
770 RETURN
800 REM CONTROLLO FORMATO PR
EZZO
810 LET X$=STR$(P)
820 IF SEG$(X$,LEN(X$)-2,1)=
CHR$(46)THEN 860
830 LET B$=".00"
840 LET X$=X$&B$
850 LET N=15-LEN(X$)
860 GOSUB 600
870 LET A$=A$&X$
880 RETURN
900 END
```

PROBLEMI

1. Esaminate il seguente programma e scrivete cosa verrà stampato, se lo si esegue.

```
100 DEF FNA(X)=2+X
110 DEF FNB(Y)=10*Y
120 DEF FNC(Z)=Z^2
130 LET R=2
140 LET S=3
150 LET T=5
160 PRINT FNC(T),FNA(S),FNB(
R)
```

```
170 LET R=S+T
180 PRINT FNA(R)+FNB(S)+FNC(
T)
190 END
```

2. Cosa darà in esecuzione quest'altro programma?

```
100 DEF FNX(A)=6*A
110 DEF FNY(B)=B+10
120 DEF FNZ(C)=C^3
130 READ P,Q,R
140 DATA 1,2,3
150 PRINT FNX(R),FNZ(P),FNY(
Q)
160 PRINT FNY(P+Q)+FNX(R)
170 END
```

3. Quale sarà il risultato di questo programma?

```
100 OPTION BASE 1
110 DIM A(5)
120 READ A(1),A(2),A(3),A(4)
,A(5)
130 DATA 6,2,7,1,3
140 GOSUB 500
150 PRINT A(1);A(2);A(3);A(4
);A(5)
160 LET A(3)=10
170 GOSUB 500
180 PRINT A(1);A(2);A(3);A(4
);A(5)
190 LET A(5)=8
200 GOSUB 500
210 PRINT A(1);A(2);A(3);A(4
);A(5)
220 STOP
500 FOR I=1 TO 4
510 LET A(I)=A(I+1)
520 NEXT I
530 RETURN
600 END
```

4. Che stampa si otterrà al seguente programma?

```
100 LET X=10
110 GOSUB 500
120 PRINT S
130 LET X=X/2
140 GOSUB 500
150 PRINT S
160 LET X=X+3
170 GOSUB 500
180 PRINT S
190 STOP
500 LET S=0
510 FOR Y=1 TO X
520 LET S=S+Y
530 NEXT Y
540 RETURN
600 END
```

5. Supponiamo che un vettore Z contenga dei numeri che devono essere sommati. Il primo elemento, Z(1), dà il numero di elementi che lo seguono. Scrivete una subroutine che cominci nella riga 800 e calcoli la somma degli elementi dopo Z(1). La somma sia assegnata alla varia-

bile T. La subroutine termini con una istruzione RETURN. Supponete che il vettore Z sia stato dimensionato opportunamente e che i valori contenuti in esso siano stati caricati nel programma principale.

6. Sia X un vettore. Il primo elemento del vettore, X(1), dà il numero di dati che lo seguono. Scrivete una subroutine che inizi alla riga 500 e cerchi il valore massimo contenuto nel vettore. Assegnate questo valore alla variabile L. Chiudete la subroutine con un RETURN. Potete supporre che il vettore X sia stato opportunamente dimensionato e riempito di numeri in qualche altra fase.

7. Scrivete un programma complementare a quello descritto nell'esempio tre. Il programma dovrebbe accettare in input record lunghi 51 caratteri da un nastro a cassetta. Assumete che il primo numero nel nastro contenga il numero di record che seguono. Dopo l'input di ogni record deve essere eseguita la sua decodifica e l'informazione deve apparire sullo schermo.

8. Supponete che un vettore Y sia riempito di numeri. Il primo elemento Y(1) dia il numero di elementi che seguono. Vogliamo una subroutine che calcoli la media (M) e la deviazione standard (S) dei numeri che seguono Y(1) nel vettore. Essa dovrebbe iniziare nella riga 900 e terminare con un RETURN. Le formule per calcolare la media e la deviazione standard sono le seguenti:

$$\text{Media} = \text{Somma dei valori} \, / \, N$$

$$\text{Deviazione standard} = \sqrt{\frac{N \times (\text{somma dei quadrati dei valori}) - (\text{somma dei valori})^2}{N \times (N-1)}}$$

ESERCIZI

Valutate i vostri progressi con i seguenti esercizi. Troverete le risposte alla fine del libro.

1. Se DEF FNA(X) = SQR(X)+3*X, Z = 2.5 e W = 10, calcolate cosa risulta:

   a. FNA(1)

   ____

b. FNA(4)

---

c. FNA(9)

---

d. FNA(Z*W)

---

2. Quale risultato si otterrà eseguendo il seguente programma?

```
100 DEF FNR(X)=X*X
110 DEF FNS(X)=3*X
120 DEF FNT(Y)=Y+1
130 LET A=1
140 PRINT FNT(A),FNR(A),FNS(
A)
150 LET M=4
160 PRINT FNR(SQR(M))
170 END
```

3. A proposito della subroutine:

a. Come si fa a passare il controllo dal programma principale alla sobroutine?

---

b. Come si ritorna dalla subroutine al programma principale?

---

c. A cosa serve l'istruzione STOP?

---

4. Cosa si otterrà in esecuzione dal seguente programma?

```
100 LET A=1
110 GOSUB 200
120 LET A=A+4
130 GOSUB 200
140 LET A=A-2
150 GOSUB 200
160 STOP
200 REM SUBROUTINE
210 IF A<2 THEN 250
220 IF A=3 THEN 270
230 PRINT "ROSSO"
```

```
240 GOTO 280
250 PRINT "BIANCO"
260 GOTO 280
270 PRINT "BLU"
280 RETURN
900 END
```

CAPITOLO DIECI

Numeri casuali e simulazioni

ARGOMENTI

Una delle applicazioni più interessanti del computer è la simulazione di eventi o processi che contengono dati casuali. Un esempio è la simulazione del gioco d'azzardo oppure la ricerca del numero di impiegati di banca richiesti per far sì che i clienti non debbano aspettare in fila più di qualche minuto, prima di essere serviti. In questo capitolo vedremo come si possono trattare problemi di questo tipo col computer. Ecco i nostri argomenti.

Caratteristiche dei generatori di numeri casuali

I computer hanno a disposizione una funzione generatrice di numeri casuali, che è il centro di tutti i programmi che riguardano dati casuali o probabilità. Studieremo come i generatori di numeri casuali possono essere impiegati in Basic.

Numeri casuali con caratteristiche particolari

Di solito si usa il generatore di numeri casuali per formare insiemi di nu-

meri casuali con caratteristiche richieste dal programmatore. Vedremo come si fa questo e come si può generare un insieme dato di numeri.

La programmazione e la simulazione

Gli esercizi pratici e i problemi di questo capitolo consisteranno in simulazioni e in applicazioni contenenti dati casuali.

PRATICA SUL CALCOLATORE

Costruire un generatore di numeri casuali

Prima di cominciare a lavorare col computer, dobbiamo parlare di alcune importanti caratteristiche dei generatori di numeri casuali. Per loro natura essi producono sequenze di numeri, che non sembrano aver relazioni o legami fra loro. Perché un generatore possa risultare utile, deve essere in grado di produrre una sequenza di numeri diversa ogni volta che lo si utilizza. Questo fatto dà luogo ad una interessante questione. Supponiamo che un programma che utilizza i numeri casuali non funzioni correttamente.
Se il problema ha a che fare con numeri casuali, sarà estremamente difficile correggerlo, perché ogni volta che si esegue il programma vengono generati numeri diversi. Per questo esiste sempre la possibilità di ripetere una data sequenza di numeri casuali ogni volta che viene eseguito il programma. Ricordate che questa caratteristica del Basic dovrà essere usata solo quando ci sono difficoltà con un programma. Col TI home computer controlliamo il tipo di sequenza di numeri casuali con la presenza o l'assenza della funzione RANDOMIZE. Se il programma contiene l'istruzione RANDOMIZE viene generata una sequenza diversa di numeri. In caso contrario viene generata una sequenza uguale.
Ed ora passiamo al lavoro sul calcolatore.

1. Accendete il computer. Se non specifichiamo altrimenti useremo l'istruzione RANDOMIZE per generare sequenze differenti di numeri casuali.

2. Scrivete il seguente programma

```
100 RANDOMIZE
110 FOR I=1 TO 10
120 PRINT RND
130 NEXT I
140 END
```

Eseguitelo e trascrivete il numero più grande e il più piccolo che sono stati stampati.

---

3. Fate funzionare il programma di nuovo. Sono apparsi gli stessi numeri?

Qual è il più grande numero stampato?

Qual è il più piccolo?

---

4. Cancellate il programma dalla memoria e scrivete il seguente:

```
100 RANDOMIZE
110 LET L=.5
120 LET S=.5
130 FOR I=1 TO 100
140 LET X=RND
150 IF X>L THEN 180
160 IF X<S THEN 200
170 GOTO 210
180 LET L=X
190 GOTO 210
200 LET S=X
210 NEXT I
220 PRINT "MAGGIORE =";L
230 PRINT "MINORE =";S
240 END
```

Questo programma esamina tutti i numeri generati dalla funzione RND e prende nota del massimo e del minimo. Così come è, il programma genera 100 numeri casuali. Eseguitelo e riportate ciò che viene stampato.

---

5. Cambiate la riga 130 con

```
130 FOR I=1 TO 1000
```

Ora il programma genererà 1000 numeri casuali. Eseguitelo e annotate quello che si ottiene.

Sulla base di ciò che avete visto finora, quale pensate sia il numero massimo che la funzione RND può generare?

Quale il più piccolo?

---

6. Passiamo a qualche altro concetto sui numeri casuali. Cancellate il programma dalla memoria e scrivete quest'altro:

```
100 RANDOMIZE
110 FOR I=1 TO 10
120 PRINT INT(2*RND)
130 NEXT I
140 END
```

Eseguitelo e riportate il risultato.

Quali erano gli unici due numeri nella stampa?

---

7. Sostituite la riga 120 con

```
120 PRINT INT(3*RND)
```

Fate una lista del programma. Quali numeri pensate che compaiano se si esegue il programma?

Fatelo e trascrivete il risultato. Potete dire qualcosa sulla sequenza o l'ordine in cui i numeri verranno stampati?

---

8. Ora cambiate la riga 120 con questa

```
120 PRINT INT(2*RND+1)
```

Cosa pensate che farà adesso il programma?

Eseguitelo e prendete nota del risultato.

---

9. Modificate ancora la riga 120

```
120 PRINT INT(4*RND+4)
```

Se si esegue il programma, secondo voi, cosa verrà stampato?

Fatelo funzionare e descrivete il risultato.

C'è qualche logica nell'output?

---

10. Bene, cambiate di nuovo la riga 120 come segue

```
120 PRINT INT(30*RND)/10
```

Fate una lista del programma ed esaminatelo attentamente. Cosa pensate che stamperà?

Eseguite il programma e riportate ciò che viene stampato.

---

11. Infine, sostituite la riga 120 con

```
120 PRINT INT(200*RND)/100
```

Listate sul video il programma. Cosa pensate che succeda eseguendolo?

Controllate se la vostra risposta è giusta. Eseguite il programma e prendete nota del risultato.

---

12. Spegnete il vostro computer. Per ora il lavoro sul calcolatore è finito.

DISCUSSIONE

Ora che avete visto al calcolatore alcune caratteristiche del generatore di numeri casuali, possiamo passare a discutere l'intero argomento con profitto.

I generatori di numeri casuali

Non ci addentreremo nei dettagli di come i numeri casuali vengano generati. Vi basti sapere che esistono parecchi metodi matematici per produrli. Il generatore di numeri casuali viene richiamato tramite la funzione RND. Essa viene usata come le altre funzioni interne del Basic che abbiamo studiato precedentemente, tranne che per due aspetti importanti. Ricordate che il risultato di una funzione dipende direttamente dal suo argomento (ciò su cui la funzione agisce). Così SQR(4) è 2, INT(3.456) è 3, e così via. Tuttavia, la funzione RND non ha argomento. Nella parte introduttiva, è stato detto che in dipendenza dell'istruzione RANDOMIZE si possono ottenere due differenti tipi di sequenze. Vale la pena di ripeterlo. Per prima cosa, se il programma contiene un'istruzione RANDOMIZE si otterrà una sequenza di numeri casuali diversa ad ogni esecuzione. Se manca l'istruzione RANDOMIZE, la sequenza che si ottiene è sempre la stessa ogni volta che si usa il programma. Questa è la differenza maggiore tra la funzione RND e le altre che abbiamo studiato.
La seconda maggiore differenza è che sembra che manchi ogni regola nel generare dei numeri con la funzione RND. Naturalmente questa è proprio la caratteristica della funzione stessa. RND significa *random*, cioè "a caso". Infatti la funzione genera a caso numeri tra 0 e 1. Tutti i numeri hanno la stessa probabilità di essere scelti. In effetti l'intervallo in cui vengono scelti i numeri va da 0.0000000000 a 0.9999999999. Lo zero uscirà molto raramente, mentre uno non potrà mai uscire.
Un buon metodo per rappresentare l'azione del generatore di numeri casuali si ha considerando la seguente situazione. Supponiamo di avere 10 miliardi di monete numerate 0.0000000000, 0.0000000001, 0.0000000002 e avanti fino a 0.9999999998 e poi 0.9999999999. Le mettiamo in un gran contenitore e le mescoliamo. Se vogliamo un numero a caso, peschiamo nel contenitore una singola moneta, leggiamo il suo numero, la rimettiamo nel contenitore e mescoliamo per bene un'altra volta. La funzione RND agisce esattamente nello stesso modo per produrre numeri casuali da usare in qualsiasi programma Basic.

Insiemi particolari di numeri casuali

Molto spesso non ci servono i numeri casuali nell'intervallo prodotto dalla funzione RND, cioè da zero a uno. Potrebbero interessarci invece i numeri interi di un certo insieme oppure i numeri casuali con delle caratteristiche particolari. Per questo dobbiamo trovare un modo per generare insiemi di numeri casuali con caratteristiche da specificare.
Vediamo quelle proprie dei numeri casuali. RND fornisce numeri da 0 a 1, quest'ultimo escluso. Se moltiplichiamo RND per N, moltiplichiamo per N tutto il risultato della funzione.
Così N*RND produrrà numeri casuali da 0 a N, escluso. Volendo, si potrebbero traslare i numeri (mantenendo uguale l'ampiezza dell'intervallo), sommando loro un altro numero. N*RND+A genererebbe numeri casuali da A ad (A+N), escluso. Infine, quando è il caso, si può prendere la parte intera di una espressione, usando la funzione INT, per ottenere numeri casuali interi.
I seguenti esempi mostrano alcuni modi di usare la funzione RND.

Espressione Basic	Risultato
5*RND+10	Numeri casuali tra 10 e 15
INT(5*RND+10)	Interi casuali 10,11,12,13,14
INT(2*RND+1)	Interi casuali 1,2
10*RND	Numeri casuali tra 0 e 100

Forse avete già incontrato le nozioni di media e di deviazione standard (vedi problema 8 nel capitolo 9). Possiamo usare la funzione RND anche per generare numeri estratti dall'insieme di quelli che hanno media e deviazione standard fissate. La regola per ottenere questi numeri è

X = M+S ((somma di 12 numeri ottenuti con RND)-6)

dove M e S sono, rispettivamente, la media e la deviazione standard. Per questa applicazione sarebbe molto utile una subroutine. Come si è detto, è come se i valori di X fossero estratti dall'insieme dei numeri con valore medio M e deviazione S. Essi potrebbero servire per simulare uno dei molti processi con distribuzione "a campana".

La correzione di programmi che usano numeri casuali

Abbiamo già accennato al fatto che il Basic fornisce un modo per eseguire più volte un programma e ripetere così la sequenza di numeri generati dalla funzione RND. È bene scrivere i programmi in modo che all'inizio generino sempre la stessa sequenza, ogni volta che li si esegue.

Quando si è sicuri del buon funzionamento del programma, lo si può modificare dandogli la possibilità di generare sequenze "a caso", sfruttando l'idea così della funzione RND.

ESEMPI DI PROGRAMMI

Esamineremo ora parecchi esempi per illustrare come si possono usare i numeri casuali. Prestate particolare attenzione ed assicuratevi di capire bene ciò che succede di volta in volta.

Esempio 1 - Testa o croce

Una delle applicazioni più semplici dei numeri casuali è la simulazione del lancio di una moneta. Vogliamo scrivere un programma che dia in esecuzione una stampa di questo tipo:

```
LANCIO    ESITO

1          T
2          C
3          C
4          T
   ecc.
```

L'esito deve essere determinato a caso da un singolo lancio della moneta, supponendo che testa e croce abbiano la stessa probabilità di uscire. Il programma dovrebbe stampare i risultati di dieci lanci.
La prima parte stampa l'intestazione e organizza lo spazio di stampa.

```
100 RANDOMIZE
110 PRINT "LANCIO","ESITO"
120 PRINT
```

Ora bisogna aprire il ciclo che conta i dieci lanci della moneta.

```
130 FOR I=1 TO 10
```

Il prossimo passo consiste nel generare casualmente 0 e 1. Supporremo che lo 0 significhi "testa" e l'1 "croce". Dovreste essere in grado di convincervi che la seguente istruzione fa proprio questo.

```
140 LET X=INT(2*RND)
```

Adesso esaminiamo X per vedere se è uscita testa (0) o croce (1).

```
150 IF X=0 THEN 180
160 PRINT I,"T"
170 GOTO 190
180 PRINT I,"C"
190 NEXT I
```

Quello che manca è solo l'istruzione END:

```
200 END
```

Ecco il programma completo:

```
100 RANDOMIZE
110 PRINT "LANCIO","ESITO"
120 PRINT
130 FOR I=1 TO 10
140 LET X=INT(2*RND)
150 IF X=0 THEN 180
160 PRINT I,"T"
170 GOTO 190
180 PRINT I,"C"
190 NEXT I
200 END
```

Questo programma si presta molto bene a mostrare come il computer possa produrre in esecuzione sequenze di numeri casuali diverse o identiche tra loro. Togliete l'istruzione RANDOMIZE per ottenere sequenze identiche.

Esempio 2 - Numeri casuali interi

Il problema che esamineremo ora consiste nello scrivere un programma Basic per generare e stampare cinquanta interi casuali compresi tra 10 e 15. L'unica parte che richiede particolare attenzione è l'istruzione per generare casualmente gli interi, perciò limiteremo a questo i nostri sforzi. Si ricordi che la funzione RND genera numeri da 0 a 1, escluso quest'ultimo. Usando la funzione INT possiamo convertire i numeri casuali in interi. INT(6*RND) fornirà a caso gli interi 0, 1, 2, 3, 4, 5. Ora è chiaro che, aggiungendo 10, otteniamo i numeri richiesti. Così, l'espressione INT(6*RND) + 10 sarà quella voluta.
Costruita questa istruzione, il programma è facilmente completato.

```
100 RANDOMIZE
110 FOR I=1 TO 50
120 LET Y=INT(6*RND)+10
130 PRINT Y,
140 NEXT I
150 END
```

Esempio 3 - Compleanni uguali

Supponiamo che cinquanta persone si trovino in una sala e che non si conoscano tra loro. Qual è la probabilità che due di loro siano nate nello stesso giorno? Consideriamo solo il giorno di nascita, non anche l'anno. Questo problema, molto noto nella teoria delle probalità, ha risultati sorprendenti. Possiamo addentrarci nel problema con la seguente strategia. Generando numeri casuali tra 1 e 365, possiamo simulare un compleanno per ogni persona. Se usiamo una matrice per elencare i compleanni nell'ordine in cui vengono generati, la ricerca dei compleanni uguali è molto semplice. Cominciando dal primo, B(1), vediamo se è uguale a uno degli altri. Poi passiamo al secondo, e così via.
In questo esempio non useremo il solito metodo, ma esamineremo subito l'intero programma, per poi spiegare i passi riga per riga.

```
100 RANDOMIZE
110 DIM B(50)
120 FOR I=1 TO 50
130 LET B(I)=INT(365*RND)+1
140 NEXT I
150 LET F=0
160 FOR I=1 TO 49
170 FOR J=I+1 TO 50
180 IF B(I)<>B(J)THEN 200
190 LET F=F+1
200 NEXT J
210 NEXT I
220 PRINT "IL NUMERO DI COPP
IE CON LO"
230 PRINT "STESSO COMPLEANNO
 E'";F
240 END
```

Ovviamente la riga 110 serve solo per dimensionare un vettore di cinquanta elementi. Le righe dalla 120 alla 140 riempiono il vettore di numeri casuali scelti tra 1 e 365, inclusi. Nella riga 150 si pone la variabile F uguale a zero. La useremo per contare il numero di coppie con lo stesso compleanno. La riga 160 apre un ciclo per individuare il compleanno che verrà confrontato con tutti gli altri. Poiché dobbiamo confrontare ognuno di essi con quelli alla sua destra, non ha senso considerare l'ultimo e quindi il valore di I si ferma a 49. Nella riga 170 si realizza la seconda parte del confronto. J inizia dal valore seguente a I e scorre tutta la lista. Nella riga 180 c'è il test per cercare il compleanno uguale. Se questo non c'è, si passa al prossimo valore di J. Se invece si è trovato un accoppiamento, con la riga 190 si incrementa di 1 il contatore delle coppie. Alla riga 220 c'è la stampa dei risultati. Possono sorgere dei problemi se tre persone sono nate nello stesso giorno. Come fareste a risolvere questa questione? È molto interessante provare questo programma. Il numero delle persone può essere modificato come si vuole e con semplici modifiche. Si può anche eseguire il programma più volte per vedere quante coppie si trovano in media in un gruppo con un numero fissato di persone.

Esempio 4 - Generatore di parole

Possiamo usare il generatore di numeri casuali per costruire delle parole. Supponiamo di dover trovare nuovi nomi per detersivi da lanciare sul mercato. Decidiamo che debbano essere formati da cinque lettere, di cui la prima, la terza e la quinta siano consonanti, mentre la seconda e la quarta siano vocali. I numeri casuali servono per estrarre le vocali dalla lista "AEIOU" e le consonanti da "BCDFGHJKLMNPQRSTVWXYZ". Scriveremo un programma Basic per far sì che il computer generi una lista di venti parole con le caratteristiche richieste. Definiamo dapprima le variabili stringa da cui pescare le vocali e le consonanti.

```
100 RANDOMIZE
110 LET A$="AEIOU"
120 LET B$="BCDFGHJKLMNPQRST
VWXYZ"
```

Avremo bisogno di interi casuali tra 1 e 5 per ottenere una vocale e di interi tra 1 e 21 per una consonante. Le istruzioni DEF si prestano ottimamente a questo socpo. L'argomento delle funzioni DEF sarà X e, poiché vogliamo ad ogni esecuzione insiemi diversi di numeri casuali, porremo X uguale a 1.

```
130 LET X=1
140 DEF FNV(X)=INT(5*RND+1)
150 DEF FNC(X)=INT(21*RND+1)
```

Ora apriamo il ciclo che genera le parole.

```
160 FOR I=1 TO 20
```

Possiamo usare le funzioni DEF per generare interi che possono essere usati di nuovo nella funzione SEG$ per pescare le lettere volute dalle stringhe A$ e B$.

```
170 LET C$=SEG$(B$,FNC(X),1)
180 LET C$=C$&SEG$(A$,FNV(X)
,1)
190 LET C$=C$&SEG$(B$,FNC(X)
,1)
200 LET C$=C$&SEG$(A$,FNV(X)
,1)
210 LET C$=C$&SEG$(B$,FNC(X)
,1)
```

La prima consonante è generata nella riga 170. Nelle righe 180, 190 e 200 vengono aggiunte una vocale, una consonante e ancora una vocale. L'ultima consonante è aggiunta nella riga 210. Il resto del programma non presenta difficoltà.

```
220 PRINT C$,
230 NEXT I
240 END
```

Ecco il programma completo:

```
100 RANDOMIZE
110 LET A$="AEIOU"
120 LET B$="BCDFGHJKLMNPQRST
VWXYZ"
130 LET X=1
140 DEF FNV(X)=INT(5*RND+1)
150 DEF FNC(X)=INT(21*RND+1)
160 FOR I=1 TO 20
170 LET C$=SEG$(B$,FNC(X),1)
180 LET C$=C$&SEG$(A$,FNV(X)
,1)
190 LET C$=C$&SEG$(B$,FNC(X)
,1)
200 LET C$=C$&SEG$(A$,FNV(X)
,1)
210 LET C$=C$&SEG$(B$,FNC(X)
,1)
220 PRINT C$,
230 NEXT I
240 END
```

Eseguitelo più volte e guardate se sono usciti i nomi dei vostri detersivi preferiti!

PROBLEMI

1. Scrivete un programma che generi e stampi venticinque numeri casuali della forma X,Y, dove X e Y sono cifre scelte a caso nell'insieme 0, 1, 2, ..., 9.

2. Scrivete un programma per generare e stampare cinquanta interi scelti a caso tra i numeri che vanno da 13 a 25.

3. Cosa verrebbe stampato dal seguente programma se lo si eseguisse?

```
100 RANDOMIZE
110 FOR N=1 TO 20
120 PRINT INT(20*RND+1)/100
130 NEXT N
140 END
```

4. Cosa darebbe, in esecuzione, questo programma?

```
100 RANDOMIZE
110 FOR I=1 TO 10
120 PRINT INT(100*RND)/10
130 NEXT I
140 END
```

5. Scrivete un programma che simuli il lancio di una moneta 10, 50, 100, 500 e 1000 volte. In ogni caso stampate il numero totale delle volte che esce "testa" e di quelle che esce "croce".

6. Costruite una simulazione del lancio dei dadi in Basic. I dadi devono essere lanciati venti volte. Per ogni lancio stampate i numeri che appaiono nelle facce superiori dei dadi.

7. Scrivete un programma che generi 1000 numeri casuali tra 0 e 1 e ne stampi la media. Quale sarà?

8. Modificate il programma dell'esempio 3 ed eseguitelo tante volte quante sono sufficienti per trovare quante persone ci vogliono perché la probabilità che almeno due di esse abbiano lo stesso compleanno sia del 50%.

9. Giovanni e Giorgio vogliono incontrarsi in biblioteca. Ognuno pensa di arrivare in biblioteca tra le 13 e le 14. Si sono messi d'accordo che ognuno di loro aspetterà 10 minuti dal momento del suo arrivo (purché non si superino le 14) e quindi, se l'altro non sarà arrivato, se ne andrà. Scrivete un programma Basic che calcoli la probabilità che Giovanni e Giorgio si incontrino.
   Realizzate una simulazione del problema che usi il generatore di numeri casuali.

10. Supponiamo che un secchio contenga palle da golf colorate. Ci sono 10 palle rosse, 5 blu, 3 verdi e 11 gialle. Scrivete un programma Basic che simuli l'estrazione a caso di 5 palle dal secchio senza che ciascuna di esse venga rimpiazzata, dopo essere stata estratta. L'output dovrebbe dare in sequenza i colori delle palle estratte.

11. Usate la regola data nel paragrafo 3 di questo capitolo per generare e stampare venticinque numeri scelti a caso tra i numeri che danno una distribuzione a campana con media 10 e deviazione standard 2. Arrotondate i numeri a 2 cifre dopo il punto decimale.

ESERCIZI

Controllate i vostri progressi risolvendo questi esercizi. Troverete le risposte alla fine del libro.

1. Scrivete un programma Basic che generi e stampi 100 numeri casuali interi scelti tra 1, 2, 3 e 4.

2. Scrivete un programma Basic che generi e stampi 100 numeri casuali tra 25 e 50.

3. Cosa darà in esecuzione il seguente programma?

```
100 RANDOMIZE
110 FOR I=1 TO 10
120 LET N=INT(2*RND+1)
130 IF N=1 THEN 160
140 PRINT "BIANCO"
150 GOTO 170
160 PRINT "ROSSO"
170 NEXT I
180 END
```

---

4. Quale sarà il risultato di questo programma?

```
100 RANDOMIZE
110 FOR J=1 TO 5
120 PRINT INT(1000*RND)/100
130 NEXT J
140 END
```

Sottoprogrammi

ARGOMENTI

Il TI 99/4A home computer è stato predisposto con la capacità di utilizzare sottoprogrammi. Questi sottoprogrammi non sono scritti in Basic ma possono essere chiamati dai programmi Basic ed anche nel modo immediato studiato nel capitolo 2. I sottoprogrammi possono essere residenti nel computer stesso oppure contenuti nei moduli di comando ''*Solid State Software*''. In questo capitolo tratteremo solo i sottoprogrammi presenti nel computer. Tuttavia, l'uso degli altri sottoprogrammi è del tutto analogo.

Manipolazione dei caratteri

Il computer contiene cinque sottoprogrammi relativi ai caratteri. Con questi sottoprogrammi i caratteri possono essere posti sullo schermo e ripetuti orizzontalmente e verticalmente, il computer può leggere quale carattere si trova in una posizione dello schermo, inoltre si possono definire nuovi caratteri.

Produzione di suoni

È possibile generare fino a tre toni contemporaneamente. Sotto il controllo di un programma Basic il computer può produrre un rumore bianco o effetti audio.

Controllo del colore

Per mezzo del sottoprogramma Colore, il computer può usufruire di una "tavolozza" con sedici colori. Usando questi colori è possibile ottenere ricchi effetti grafici.

Definizione della tastiera

Spesso è importante che il computer scopra direttamente quanto è successo sulla tastiera, soprattutto nei programmi educativi e didattici. Un sottoprogramma provvede alla programmazione dei tasti per le risposte.

PRATICA SUL CALCOLATORE

Nella parte seguente ci riferiremo al set di caratteri in codici ASCII. Mentre fate pratica sul calcolatore dovreste avere a disposizione il codice dei caratteri ASCII contenuto nel manuale del TI home computer.
Iniziamo ora la pratica al calcolatore.

1. Accendete il computer, selezionate il Basic e scrivete il seguente programma:

```
100 INPUT "R = ":R
110 INPUT "C = ":C
120 INPUT "N = ":N
130 INPUT "M = ":M
140 CALL CLEAR
150 CALL HCHAR(R,C,N,M)
160 END
```

   Date il RUN ed inserite i valori 10, 10, 72 e 5 rispettivamente per R, C, N ed M. Che cosa è successo?

   ____

2. Eseguite il programma tre volte, ponendo R uguale a 10, 15 e 20.

Assegnate alle altre variabili gli stessi valori che avevano al punto 1. A quale parte dello schermo sembra riferirsi la grandezza R?

---

3. Bene, ora tenete R uguale a 10, ma eseguite il programma ponendo C uguale a 10, 15 e 20. Attribuite ad N e ad M i valori 72 e 5. A quale elemento dello schermo si riferisce la variabile C?

---

4. Ora che abbiamo visto la funzione di R e C nell'istruzione CALL HCHAR, fissiamo la nostra attenzione alla parte svolta da N. Ponete R, C ed M uguali a 10, 10 e 5 ma eseguite il programma usando valori di N variabili nell'intervallo da 48 a 90. Che cosa controlla la N?

---

5. Ora date ad R, C ed N i valori 10, 10 e 72 rispettivamente. Eseguite il programma con M uguale a 10, 20 e 50. Cosa controlla la M?

A cosa si riferisce il prefisso H in CALL HCHAR?

---

6. Ora che abbiamo sperimentato il sottoprogramma HCHAR, passiamo ad una nuova funzione. Cambiate la linea 150 in

```
150 CALL VCHAR(R,C,N,M)
```

Vi sarà più facile da capire il sottoprogramma VCHAR dopo aver fatto esperienza con HCHAR. Date il RUN diverse volte cambiando i valori di R, C, N ed M. Cosa controlla la R?

Cosa controlla la C?

Cambiando la N che cosa cambia?

A che cosa serve M nell'istruzione VCHAR?

Cosa significa V in VCHAR?

---

7. Cancellate il programma in memoria e scrivete il seguente:

```
100 CALL CLEAR
110 CALL HCHAR(5,5,65)
120 CALL HCHAR(6,6,66)
130 CALL HCHAR(7,7,67)
140 CALL HCHAR(8,8,68)
150 INPUT "R = ":R
160 INPUR"C = ":C
170 LET C=C-2
180 CALL GCHAR(R,C,N)
190 PRINT "IL CARATTERE CHE
SI TROVA"
200 PRINT "NELLA POSIZIONE D
ATA E' ";CHR$(N)
210 END
```

Studiate il programma per qualche momento. La novità sta nel sottoprogramma GCHAR alla linea 180. Eseguite il programma ed assegnate il valore 6 sia ad R che a C. Cos'è successo?

---

8. Bene, se date il RUN ed inserite il valore 8 per R e C, cosa accadrà?

Provate e segnate qui sotto cosa succede.

---

9. Eseguite ora il programma dando il valore 10 ad R e 15 a C. Cosa è successo?

Cosa c'è sullo schermo in corrispondenza di R = 10 e C = 15?

---

10. Da questi tentativi dovreste intuire cosa fa il comando GCHAR. In particolare, nell'espressione GCHAR(R,C,N) a cosa si riferiscono R e C?

Quale funzione svolge N?

Cosa sta ad indicare la G?

Se avete saputo rispondere alle precedenti domande, bene. Altrimenti non preoccupatevi poiché torneremo sugli stessi concetti nella discussione.

11. Cancellate il programma in memoria e consideriamo un altro sottoprogramma. Inserite il seguente programma:

```
100 CALL CLEAR
110 INPUT "SCRIVI LA STRINGA
":A$
120 CALL CHAR(128,A$)
130 CALL HCHAR(15,15,128)
140 END
```

Il CALL CLEAR alla linea 100 ci è già familiare poiché l'uso di questo comando è stato richiesto più volte in precedenza. La stringa richiesta in input alla linea 110 è usata nel nuovo sottoprogramma CHAR alla linea 120. E ancora, il numero 96 che appare nell'istruzione CHAR è usato nel sottoprogramma HCHAR alla linea 130. Avviate il programma e, al segnale di input, scrivete la stringa FF83858991A1C1FF, premete quindi il tasto ENTER. Cosa è successo?

Il carattere al centro dello schermo appartiene a quelli in codice ASCII?

---

12. Bene, eseguite nuovamente il programma e questa volta inserite i sedici caratteri 30468991523C1010 (il 2°, il 14° e il 16° sono degli zero), e premete ENTER. Cos'è successo?

Sul video dovreste vedere una lettera greca (non appartenente al codice ASCII). A cosa serve il sottoprogramma CHAR?

---

13. Soltanto una volta ancora. Eseguite il programma usando la stringa 0F03050810204080. In questa stringa il carattere più usato è uno zero non la lettera O. Cos'è successo?

---

14. Ora listate il programma e studiatelo brevemente. Naturalmente la stringa A$ che inseriamo controlla il nuovo carattere nella funzione CHAR. Quale parte compete al numero 96 usato nelle istruzioni CHAR e HCHAR?

Per ora dovete accontentarvi di sapere che possono essere creati nuovi caratteri. Più avanti l'argomento verrà esaminato a fondo e imparerete a disegnare tutti i caratteri che vorrete.

15. Passiamo ora ad un'altra istruzione. Cancellate il programma in memoria e scrivete quello seguente:

```
100 CALL CLEAR
110 INPUT "DURATA":D
120 INPUT "TONO":T
130 INPUT "INTENSITA'":I
140 CALL SOUND(D,T,I)
150 END
```

Dovrebbe essere chiaro che questo sottoprogramma produce un suono. Assicuratevi che il volume del vostro apparecchio televisivo sia regolato in maniera opportuna prima di continuare. Ora eseguite il programma. Ponete la DURATA uguale a 1000, il TONO uguale a 264 e il VOLUME uguale a zero. Cosa è accaduto?

---

16. Ora eseguite il programma più volte lasciando DURATA e TONO uguali a 1000 e 264 ma cambiando il VOLUME a 5, 10, 15 e 20. Quando aumenta il numero associato al volume, come varia il volume stesso?

Per ora la cosa vi può sembrare poco chiara. La ragione vi verrà spiegata più avanti, non preoccupatevi!

17. Eseguite il programma diverse volte. Adesso ponete la DURATA uguale a 1000 e il VOLUME uguale a zero ma date al TONO i valori

264, 297, 330, 352, 396, 440, 495 e 528. Cosa controlla la variabile TONO?

Quando il numero assegnato al TONO aumenta, cosa accade all'altezza del suono emesso?

---

18. Ora lasciate TONO e VOLUME ai valori 264 e 0 ma cambiate la DURATA. Eseguite il programma con la DURATA uguale a 4000, 2000, 1000, 500 e 250. Cosa controlla la DURATA?

Quali caratteristiche ha il suono quando il numero assegnato alla DURATA diventa più piccolo?

---

19. Facciamo ancora una prova prima di lasciare il programma SOUND. Cancellate il programma dalla memoria. Invece di usare un altro programma lavoreremo in modo immediato. Scrivete il seguente comando:

```
CALL SOUND(100,264,0)
```

Cos'è successo?

Non dovrebbe esserci alcuna sorpresa poiché abbiamo appena affrontato questo tipo di problema.

20. Scrivete il seguente comando:

```
CALL SOUND(1000,264,0,330,0)
```

Cosa fa il computer ora?

---

21. Provate ognuno dei seguenti comandi CALL. Di volta in volta dovreste essere in grado di prevedere cosa accadrà.

```
CALL SOUND(100,264,0,330,0,3
96,0)
CALL SOUND(1000,264,0,352,0,
440,0)
CALL SOUND(1000,264,0,528,0)
```

22. Ora tratteremo un nuovo sottoprogramma. Scrivete le seguenti istruzioni:

```
100 CALL CLEAR
110 LET A$="ABCDEFGHIJKLMNOP
QRSTUVWXYZ"
120 PRINT A$
130 LET C=16
140 FOR GRUPPO=5 TO 6
150 FOR COLORE=2 TO 15
160 CALL COLOR(GRUPPO,COLORE
,C)
170 FOR TEMPO=1 TO 100
180 LET X=1
190 NEXT TEMPO
200 NEXT COLORE
210 NEXT GRUPPO
220 END
```

Non stupitevi per il ciclo della variabile PAUSA alle linee 170, 180 e 190. Serve ad una più lenta esecuzione del programma. Cosa è accaduto?

____

23. Ora cambiate nel modo seguente la linea 160

```
160 CALL COLOR(GRUPPO,COLORE
,COLORE)
```

Eseguite il programma e descrivete quanto è successo.

____

A questo punto va anche detto che può essere cambiato il colore dello schermo. Poiché una spiegazione completa del sottoprogramma COLOR è piuttosto complicata, non cercheremo di approfondire oltre l'argomento. Verrà ripreso dettagliatamente nella discussione e negli esempi che seguiranno.

24. Questo conclude la parte pratica. Spegnete il computer e passate al paragrafo successivo.

DISCUSSIONE

Dopo aver visto cosa si può ottenere con i sottoprogrammi, dovreste essere maggiormente orgogliosi del vostro TI 99/4A home computer! È importante tuttavia tornare sull'argomento e considerarlo nei dettagli.

Manipolazione dei caratteri

Prima di inoltrarci nella discussione dei sottoprogrammi relativi ai caratteri, dovremmo rivedere brevemente il set di caratteri ASCII. Troverete una sua completa descrizione sul manuale. A noi serve esaminare solo certe parti. Per prima cosa esistono 128 caratteri. Possiamo usare la funzione CHR$(N) per avere il carattere a partire dal suo numero di codice. Molti dei caratteri ASCII non rivestono alcuna importanza nella nostra discussione. Dobbiamo limitare il nostro interesse a quelli con numero compreso tra il 32 e il 95.

Il carattere 32 è lo spazio. Ciò è importante poiché, quando scrivete CALL CLEAR, dite al computer di riempire lo schermo di caratteri numero 32, cioè di riempirlo di spazi vuoti. Più avanti torneremo su questo punto.

I caratteri compresi negli intervalli 33-47, 58-64 e 91-95 sono usati nella punteggiatura e nella notazione matematica. Potete controllarli nella tavola dei caratteri ASCII nel manuale del vostro computer. Le cifre da 0 a 9 hanno numeri di codice ASCII dal 48 al 57. Le lettere maiuscole A-Z hanno numeri dal 65 al 90. Le lettere minuscole a-z hanno numeri di codice ASCII dal 97 al 123. Vediamo ora come si possono definire nuovi caratteri o simboli. Ogni carattere dello schermo è formato da una matrice di punti avente otto righe ed otto colonne. Ogni punto di questa matrice può essere attivato o lasciato vuoto. L'insieme di punti attivati genera il carattere.

Nei normali caratteri è utilizzata solo la parte centrale di sei punti per sei. Il bordo lasciato vuoto provvede alla giusta separazione dei caratteri sia in senso orizzontale che verticale. Nella creazione dei caratteri è anche possibile definire il colore ma questo è un punto che considereremo più avanti.

Qui sotto viene riportato un esempio di come una matrice di punti possa generare un simbolo. Le X rappresentano i punti e le 0 gli spazi vuoti.

```
0 0 0 X X 0 0 0
0 0 0 X X 0 0 0
0 0 0 X X 0 0 0
X X X X X X X X
```

```
X X X X X X X X
0 0 0 X X 0 0 0
0 0 0 X X 0 0 0
0 0 0 X X 0 0 0
```

Questo insieme di X e 0 occupa tutte le otto per otto posizioni e definisce il segno più, ma, naturalmente, utilizzando in vario modo tutte le 64 posizioni possiamo disegnare ciò che desideriamo. La disposizione dei punti va comunicata al computer quattro posizioni alla volta, due gruppi di quattro posizioni per ogni riga. Nel nostro esempio, alla riga 1, i due gruppi sono 000X e X000.
Per descrivere un blocco di quattro punti viene usato un solo carattere. Per far questo si può seguire la seguente tabella:

CARATTERE	PUNTI	CARATTERE	PUNTI
0	0000	8	X000
1	000X	9	X00X
2	00X0	A	X0X0
3	00XX	B	X0XX
4	0X00	C	XX00
5	0X0X	D	XX0X
6	0XX0	E	XXX0
7	0XXX	F	XXXX

Allora la nostra disposizione di punti 000X e X000 sarà rappresentata dai caratteri 8 e 1. Poiché ogni riga viene definita da due caratteri, e ci sono otto righe, sedici caratteri definiranno la matrice di punti completa. Il segno più dell'esempio precedente è definito dalla stringa 818181FFFF818181.
Per definire i vostri simboli è consigliabile usare un foglio quadrettato. Segnate un quadrato di 8 per 8 quadratini e annerite quelli necessari per creare il nuovo carattere. Dopo aver fatto questo, scrivete i caratteri in codice che definiscono i quattro punti a sinistra e a destra di ogni riga. Ora la stringa di sedici caratteri necessaria a definire il nuovo simbolo può essere letta facilmente.
Ad esempio, immaginiamo di voler rappresentare il simbolo della lettera greca minuscola lambda. La matrice di punti che definisce tale lettera è riportata qui sotto.

```
E    X X X 0 0 0 0 0    0
3    0 0 X X 0 0 0 0    0
1    0 0 0 X X 0 0 0    8
0    0 0 0 0 X X 0 0    C
1    0 0 0 X X X X 0    E
3    0 0 X X 0 0 X X    3
6    0 X X 0 0 0 0 X    1
C    X X 0 0 0 0 0 X    1
```

La stringa di caratteri per definire la lettera lambda è E030180C1E3361C1. Dobbiamo ora memorizzare nel computer il nostro nuovo simbolo. Se decidiamo di assegnare al nuovo simbolo (lambda) il numero di carattere 96, dobbiamo usare l'istruzione CALL CHAR(96,A$) dove A$ = E030180C1E3361C1. In seguito, facendo riferimento al numero di carattere 96, potremo ottenere l'apparizione del nostro nuovo carattere. Potremmo proseguire con la definizione di nuovi caratteri attribuendo loro i numeri 97, 98, 99 e così via.
Prima di lasciare questo sottoprogramma è necessario considerare alcuni punti importanti. È possibile, anche se probabilmente non consigliabile, ridefinire i caratteri con i numeri inferiori al 96. Supponete di aver ridefinito il carattere 32 con qualche altro nuovo simbolo e immaginate che il programma incontri poi un'istruzione CALL CLEAR. Naturalmente, poiché il CALL CLEAR riempie lo schermo di caratteri numero 32, il vostro video verrà riempito tutto con il nuovo simbolo invece di venir cancellato come vi aspettavate. Quando il programma giunge a termine o viene interrotto, i caratteri con numero inferiore al 128 tornano alla loro definizione originale. Di conseguenza, se, dopo aver visto lo schermo pieno di "cose strane", fermate il programma e scrivete CALL CLEAR, allora otterrete il consueto schermo pulito.
Il punto è che la ridefinizione dei caratteri al di sotto del numero 128 può produrre effetti strani ed imprevisti. Pertanto siate prudenti nell'attribuire ai nuovi caratteri numeri inferiori a 128. E questo per due ragioni. Primo, evitate così di disturbare il set di caratteri che viene normalmente usato. Secondo, la definizione dei caratteri con numero superiore a 128 non va perduta quando il programma viene interrotto o finisce. Essi vanno persi solo quando il computer viene spento o quando si scrive NEW. Se lo desiderate, l'intero set di caratteri può essere utilizzato nel sottoprogramma CALL CHAR.
Restano da discutere altri tre sottoprogrammi, sempre in relazione alla manipolazione dei caratteri. Il primo di questi, il sottoprogramma CALL GCHAR, può essere spiegato molto facilmente. Lo scopo di questo sottoprogramma è quello di identificare un carattere sullo schermo. Un esempio è

```
CALL GCHAR(10,12,N)
```

Quando viene eseguito, alla variabile N sarà assegnato il numero di codice ASCII del carattere situato alla riga 10, contando dall'alto, e alla colonna 12, contando da sinistra. Se in questa posizione si fosse trovata la lettera A, N avrebbe avuto il valore 65 (il numero di codice ASCII della lettera A). Come argomento dell'istruzione CALL GCHAR potete usare qualsiasi nome di variabile numerica. Così,

```
150 CALL GCHAR(Y,X,C)
160 CALL GCHAR(RIGA,COL,M)
```

rappresentano istruzioni valide. Ricordate che le tre variabili numeriche usate nel sottoprogramma GCHAR hanno significati specifici. La prima di questo indica al computer la riga, a partire dalla parte alta dello schermo. La seconda specifica il numero di colonna, partendo dalla sinistra dello schermo. Questi due numeri (coordinate) individuano una posizione sullo schermo. Il numero di codice ASCII del carattere situato in tale posizione è assegnato alla terza variabile numerica presente nella funzione GCHAR.
Talvolta il video lascia libere le colonne ai margini sinistro e destro. Conseguentemente, le colonne 1, 2, 31, e 32 non saranno visibili sullo schermo.
Si possono tracciare linee orizzontali sul monitor con il sottoprogramma CALL HCHAR. Una tipica istruzione è

```
100 CALL HCHAR(Y,X,N,R)
```

Questa istruzione dice al computer di iniziare una linea orizzontale alla riga Y, contata partendo dal limite superiore dello schermo, e alla colonna X, da sinistra. N si riferisce al numero di codice ASCII del carattere che formerà la linea. Notate che questo comprende anche i caratteri speciali definiti con il sottoprogramma CALL CHAR e aventi numeri superiori a 128. Qualsiasi carattere venga usato, esso viene ripetuto R volte a partire dalla posizione individuata da X ed Y. Se R è troppo grande e tale che la linea supera il margine destro dello schermo, essa viene completata dal margine sinistro della riga successiva. Non è indispensabile includere l'argomento R. Ad esempio,

```
100 CALL HCHAR(Y,X,N)
```

va bene. Poiché non viene specificato il numero delle ripetizioni, il computer stampa un solo carattere (definito da N) e lo pone nella posizione X e Y.
In modo analogo vengono tracciate linee verticali. L'istruzione

```
100 CALL VCHAR(Y,X,N,R)
```

produce una linea verticale composta da caratteri aventi numero di codice ASCII corrispondente ad N, a partire dalla posizione X e Y. Comunque, questa linea sarà ottenuta sullo schermo con la ripetizione del carattere R volte. Se la linea supera il limite inferiore dello schermo, essa viene completata alla sommità del video, una colonna più a destra.

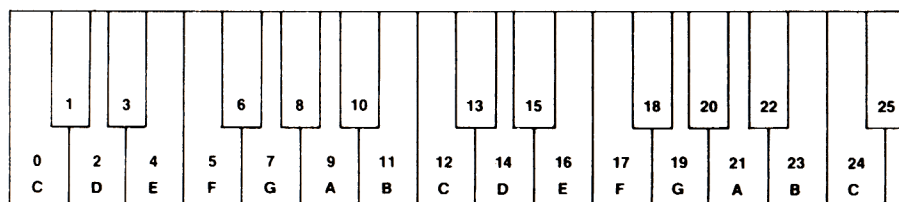
Produzione di suoni

Con il sottoprogramma CALL SOUND potete generare note udibili attraverso l'apparecchio televiso. Una generica istruzione è

```
100 CALL SOUND(D,P,A)
```

In questa funzione, l'argomento D definisce la durata del suono in millisecondi. Poiché in un secondo ci sono 1000 millisecondi, ponendo D uguale a 1000, specificherete per la nota una durata di un secondo. D può avere un valore massimo di 4275 (poco più di quattro secondi) fino ad un minimo di 1. Se in un programma usate CALL SOUND con D = 1 (1 millesimo di secondo), il suono verrà udito facilmente e certamente sarà più lungo di un millesimo di secondo. Ciò accade perché la specificazione di durata da 1 a 4275 si riferisce ai particolari microcircuiti del computer adibiti alla generazione del suono. Tuttavia, le caratteristiche del programma, combinate a quelle dei circuiti audio e dell'altoparlante producono suoni udibili per una frazione di tempo diversa da quella specificata. Questo effetto è particolarmente pronunciato per valori al di sotto di D = 100.
Quando viene posta un'istruzione CALL SOUND in un programma, il computer la passa al generatore di suoni e poi passa all'istruzione successiva del programma. Supponete che venga incontrata un'altra istruzione CALL SOUND mentre si sta ancora producendo il suono precedente. Ciò che accadrà dipende dal segno algebrico che precede il valore di durata nella seconda istruzione CALL SOUND. Se la durata è positiva, il computer aspetta che il primo suono venga terminato, poi produrrà il secondo. Se il valore è negativo, il primo suono è interrotto immediatamente e viene generato il secondo.
Ancora, tornando alle specifiche del sottoprogramma SOUND, il secondo numero dà l'altezza (o frequenza) del suono in cicli per secondo. P può avere il valore minimo di 110 fino al limite dell'udibilità (circa 20000 cicli al secondo) come massimo. Per darvi un'idea di valori ragionevoli per le frequenze, in un pianoforte il valore mediano corrisponde ad una frequenza pari a 264. La stessa nota dell'ottava superiore corrisponde a 528 e nell'ottava inferiore, a 132. Allora le frequenze della parte centrale della tastiera del piano vanno da 132 a 528.
L'ultimo numero che appare nella funzione SOUND controlla l'ampiezza del suono. Precisamente, questo numero dà l'attenuazione in db (decibel) che il computer applicherà alla nota. Se A è 0, non c'è alcun attenuazione, e il risultato è un suono a maggior volume. Un valore di A pari a 30 corrisponde ad una attenuazione di 40 db ed il suono risulta al volume minimo. Si possono specificare tutti i valori da 0 a 30.
È possibile specificare fino a tre note allo stesso tempo. Il modo è il seguente

```
100 CALL SOUND(D,P1,A1,P2,A2
,P3,A3)
```

La stessa durata D è applicata ai tre toni. Il primo tono ha altezza P1 e attenuazione A1. Il secondo tono è definito da P2 ed A2, il terzo da P3 e A3. Se si desidera ottenere solo due toni, P3 e A3 vanno omessi dalle specifiche SOUND.
Si possono anche ottenere rumori ed effetti speciali dando alla frequenza un valore negativo, da -1 a -8. Per maggiori dettagli consultate il vostro manuale.
Prima di lasciare il sottoprogramma CALL SOUND, dovete conoscere un po' di teoria musicale se volete ottenere musica. La scala musicale comunemente usata nella civiltà occidentale contiene dodici note. Lo sviluppo di questa scala richiese lungo tempo e generò innumerevoli controversie le quali furono più o meno risolte da Johann Sebastian Bach nel suo *Il clavicembalo ben temperato*. In questi ventiquattro brani Bach dimostrò il valore di una scala "temperata", cioè la scala con dodici note che usiamo ai nostri giorni.
In ogni caso, è possibile stabilire questa scala matematicamente. Seguiremo questa via riferendoci allo schema della tastiera di un pianoforte riportato qui sotto.

$$\begin{aligned} freq &= f_0(2^{n/12}) \\ &= f_0(2^{1/12})^n \\ &= f_0(1.059463094)^n \end{aligned}$$

l'esponente n è il numero scritto su ogni tasto

In questo diagramma tutto viene riferito al tasto C di sinistra che ha una frequenza di 264 cicli al secondo. I tasti sono numerati partendo dallo zero (do centrale), il numero 12 è il do di un'ottava superiore, e così via. La scala principale (tasti bianchi) corrisponde ai numeri 0, 2, 4, 5, 7, 9, 11 e 12. Questa scala è formata dalle note familiari do, re, mi, fa, sol, la, si. La scala cromatica include tasti bianchi e neri corrispondenti ai numeri 0, 1, ..., 11 e 12.
La parte fondamentale di questa discussione è che possiamo specificare i toni in base ai numeri dei tasti lasciando al computer il compito di stabilire la giusta frequenza. Per il tasto N la corretta frequenza è

```
NOTA=(FREQ)*(1.059463094)^N
```

Normalmente poniamo la frequenza base al do centrale (264 cicli al secondo). Questo, tuttavia, non è indispensabile e un brano musicale suonato al computer può essere suonato in modo diverso semplicemente cambiando la frequenza base. Il numero 1.059463094 è la radice dodicesima di 2. Così, se vogliamo la frequenza della nota 12, la radice dodicesima di 2 alla dodicesima potenza è uguale a 2, e il doppio della frequenza base dà la frequenza della stessa nota nell'ottava superiore.
Per N avente i valori riportati nella scala principale, possiamo calcolare i dati seguenti:

Tasto	Nota	N	Frequenza
C	do	0	264,00
D	re	2	296,33
E	mi	4	333,62
F	fa	5	352,40
G	sol	7	395,55
A	la	9	443,99
B	si	11	499,37
C	do'	12	528,00

Non è necessario usare questi dati. La tabella è stata riportata qui solo per dimostrare come la formula sia corretta per definire le frequenze della scala principale. Col computer useremo i numeri scritti sui tasti del pianoforte e lasceremo che sia esso a generare le giuste frequenze. Questo verrà dimostrato negli esempi.
Per inciso, va specificato che esistono molte scale musicali compresa quella orientale con quattro note, una medio-orientale con otto note e variazioni della scala occidentale con dodici note. Il TI 99/4A home computer fornisce un modo potente e flessibile per sperimentare la musica di differenti culture. I testi di teoria musicale riportano le informazioni necessarie per creare le differenti scale al computer.

Controllo del colore

Il colore è controllato con il sottoprogramma COLOR. Una semplice istruzione di questo tipo è

```
100 CALL COLOR(A,B,C)
```

Per gli argomenti della funzione COLOR si possono usare tutti i nomi di variabile numerica che si desiderano. Qui abbiamo scelto i nomi A, B e C. Il primo argomento (A) definisce l'insieme al quale appartiene il carattere in questione. Affronteremo questo punto più tardi. B specifica il colore dei punti che formano il carattere e C specifica il colore dello

sfondo del carattere. Il computer può generare sedici differenti colori selezionabili con un numero. I colori e i rispettivi numeri sono riportati nella seguente tabella:

Colore	**Numero**	**Colore**	**Numero**
Trasparente	1	Rosso	9
Nero	2	Rosso chiaro	10
Verde	3	Giallo scuro	11
Verde chiaro	4	Giallo chiaro	12
Blu scuro	5	Verde scuro	13
Blu chiaro	6	Magenta	14
Rosso scuro	7	Grigio	15
Ciano	8	Bianco	16

I caratteri che rientrano nell'istruzione CALL COLOR sono riuniti in sottoinsiemi del set di caratteri ASCII più i caratteri speciali definiti dall'utente. Ognuno di questi gruppi viene definito da un numero. La tabella qui sotto mostra questa relazione.

Insieme	**N**	**Insieme**	**N**
1	32-39	9	96-103
2	40-47	10	104-111
3	48-55	11	112-119
4	56-63	12	120-127
5	64-71	13	128-135
6	72-79	14	136-143
7	80-87	15	144-151
8	89-95	16	152-159

Prima di vedere come cambiare i colori dei vari insiemi, rivediamo cosa si verifica in ogni apparecchio televisivo. L'intera immagine viene ricreata (o ridisegnata se volete) trenta volte al secondo. Se, per esempio, sullo schermo appare la lettera Z, ad ogni secondo essa deve essere ricreata trenta volte. Le informazioni sul modo di disegnare la Z devono provenire da qualche parte. Per un normale apparecchio TV, tali informazioni provengono dalle stazioni televisive sotto forma di onde radio. Tuttavia, con il computer, l'immagine si forma in modo differente. Nella memoria sono collocate le "mappe di punti" che costituiscono ogni carattere. Studiando il sottoprogramma CALL CHAR vedemmo che la mappa dei punti che formano un simbolo o un carattere è una stringa lunga sedici caratteri. Sono richieste ulteriori informazioni per dire al computer i colori da usare. Tutte queste informazioni vengono lette dalla memoria trenta volte al secondo per mantenere l'immagine sullo schermo. Se cambiate le istruzioni per creare caratteri, allora immediatamente cambierà l'immagine di tutti i caratteri coinvolti.

Notate che ogni insieme contiene otto caratteri. Per vedere quali caratteri sono contenuti in ogni gruppo consultate la tabella dei codici ASCII del vostro manuale. Diremo solo dove si trovano alcuni dei caratteri. Le lettere maiuscole A-G sono nell'insieme 5, H-O nell'insieme 6, P-W nel numero 7 e X-Z nell'8. Le cifre 0-9 si trovano nei gruppi 3 e 4.
Vediamo ora come si usa il sottoprogramma COLOR. L'istruzione

```
100 CALL COLOR(5,9,12)
```

cambia le informazioni relative alle lettere A, B, C, D, E, F, e G (insieme numero 5). I punti che formano le lettere appariranno rossi (colore numero 9), e lo sfondo di ogni lettera sarà giallo chiaro (colore numero 12). Quando questa istruzione verrà eseguita dal computer, tutti i caratteri dalla A alla G sullo schermo assumeranno immediatamente i nuovi colori.
Se vogliamo cambiare i colori di tutto l'alfabeto e delle cifre numeriche, dobbiamo scrivere tante istruzioni CALL COLOR quanti sono gli insiemi dal 3 all'8. Notate che il gruppo 13 inizia con il carattere numero 128, il primo disponibile per la definizione di nuovi caratteri con il sottoprogramma CALL CHAR.
Resta da fare un'ultima precisazione sul colore del video. Come abbiamo già detto, i colori dei caratteri vengono determinati specificando il colore del carattere stesso e dello sfondo (cioè la parte della matrice di otto per otto punti non occupata dal carattere). Tuttavia, entrambi questi "strati" di colore stanno su un terzo strato, il colore di fondo dello schermo. Questo colore può essere controllato con il sottoprogramma CALL SCREEN. Un esempio potrebbe essere

```
280 CALL SCREEN(N)
```

dove N contiene un numero tra 1 e 16, estremi compresi. Naturalmente questi sedici numeri sono i codici di colore sui quali abbiamo già riferito. Si possono ottenere effetti molto interessanti sul video. Ad esempio, se si sceglie come colore di fondo il blu chiaro (codice colore 6), si lascia trasparente lo sfondo del carattere (colore 1) e si rende bianco il carattere stesso (colore 16), si ottengono, come risultato, scritte bianche su uno sfondo azzurro. Usando i sottoprogrammi CALL CHAR, CALL COLOR e CALL SCREEN si possono ottenere sullo schermo molti effetti interessanti da un punto di vesta cromatico.

Controllo della tastiera

Questo argomento è piuttosto specializzato e non è stato considerato nelle attività pratiche. Tuttavia ci sono casi in cui è necessario interroga-

re la tastiera sotto controllo del programma. Si ottiene questo risultato con il sottoprogramma CALL KEY. Una generica istruzione è

```
100 CALL KEY(0,N,S)
```

La prima variabile è posta uguale a zero. Ciò sta a significare che si interroga la tastiera. Per scopi particolari si usano codici diversi. Per maggiori dettagli consultate il manuale del computer.
L'ultima variabile (S) riflette lo stato della tastiera. Se, dopo che è stata eseguita l'ultima istruzione CALL KEY, viene premuto un nuovo tasto, S prende il valore +1. Se viene premuto lo stesso tasto che era stato premuto nell'ultima CALL KEY, allora S sarà -1. Infine, se non viene premuto alcun tasto, S assume il valore 0. Pertanto, in base al valore di questa variabile (si può usare qualsiasi nome di variabile numerica), possiamo sapere cosa è accaduto sulla tastiera.
Se l'indicatore dello stato della tastiera è +1 o -1, il secondo argomento (N) contiene il numero del carattere impresso sul tasto premuto durante l'esecuzione dell'istruzione CALL. Questo può essere trasformato nel carattere corrispondente con la funzione CHR$(N).
Il sottoprogramma CALL KEY vi permette di sapere se viene premuto un tasto oppure no e, nel caso affermativo, di sapere quale tasto è stato premuto. Poiché tale controllo avviene attraverso un programma Basic, le capacità del vostro computer assumono una dimensione diversa.

ESEMPI DI PROGRAMMI

Ora vedremo alcuni programmi che sfruttano vantaggiosamente i sottoprogrammi appena discussi.

Esempio 1 - "Fra Martino"

Per mostrare le capacità musicali del TI 99/4A home computer scriveremo un programma per suonare un pezzo a tre voci. Poiché la melodia di "Fra Martino" è familiare a molti, abbiamo scelto questo motivo.
Useremo variabili con indici per memorizzare le note che verranno suonate. Così, specifichiamo la *option base* e dimensioniamo gli indici.

```
100 OPTION BASE 0
110 DIM S(26),K(65,3)
```

Il nostro obiettivo sarà quello di comporre la scala nel vettore S che conterrà ventisei frequenze, corrispondenti a quelle della tastiera consi-

derata nella discussione. Il vettore K servirà a memorizzare quali tasti si suppone di "premere" di volta in volta. La matrice K ha una dimensione uguale a 65 ma ignoreremo i valori di ordine zero. Nella musica ogni valore di K corrisponde ad una nota da un ottavo. Se vogliamo una nota da un quarto, dobbiamo ripetere due volte un ottavo. Una nota da un mezzo si ottiene con quattro ottavi, e così via. La canzone "Fra Martino" richiede 64 note da un ottavo per generare, nell'esecuzione musicale, note da una metà, un quarto ed un ottavo.
Per prima cosa creiamo la scala.

```
120 LET FREQ=264
130 FOR N=0 TO 25
140 LET S(N)=FREQ*1.05946309
4^N
150 NEXT N
```

Tutte le frequenze sono calcolate con riferimento alla frequenza base di 264 cicli al secondo (do centrale). Faremo in modo che l'indice della matrice S corrisponda ai numeri dei tasti riportati sul disegno della tastiera. Il tasto 0 è associato a S(0) che ha il valore 264. Il tasto 12 corrisponde a S(12) che ha il valore 528, e così via.
Ora creiamo un ciclo che permetta l'inserimento dei tre numeri dei tasti per ognuna delle 64 note da un ottavo che costituiscono la musica.

```
160 FOR R=1 TO 64
170 PRINT R
180 INPUT K(R,1),K(R,2),K(R,
3)
190 NEXT R
```

Abbiamo scritto la variabile di riga R per indicare quale gruppo di tre numeri deve essere inserito. I dati da introdurre verranno forniti più avanti.
Ora dobbiamo suonare la musica.

```
200 LET D=250
210 FOR R=1 TO 64
220 CALL SOUND(D,S(K(R,1)),0
,S(K(R,2)),0,S(K(R,3)),0)
230 NEXT R
240 GOTO 210
250 END
```

Nella linea 200 abbiamo posto la durata di una nota da un ottavo pari a 250 millisecondi (un quarto di secondo). Poi dovremo ripetere il ciclo relativo alla matrice K inserendo di volta in volta tre numeri corrispondenti ad altrettanti tasti; questi verranno utilizzati nel vettore di scala S per ottenere le frequenze necessarie nell'istruzione CALL SOUND.
Riportiamo di seguito il programma completo:

```
100 OPTION BASE 0
110 DIM S(26),K(65,3)
120 LET FREQ=264
130 FOR N=0 TO 25
140 LET S(N)=FREQ*1.05946309
4^N
150 NEXT N
160 FOR R=1 TO 64
170 PRINT R
180 INPUT K(R,1),K(R,2),K(R,
3)
190 NEXT R
200 LET D=250
210 FOR R=1 TO 64
220 CALL SOUND(D,S(K(R,1)),0
,S(K(R,2)),0,S(K(R,3)),0)
230 NEXT R
240 GOTO 210
250 END
```

In questo esempio abbiamo voluto creare l'esecuzione delle tre "voci" contemporaneamente, ognuna al suo posto nella melodia. Ci vorrà un po' di tempo per scrivere tutti i numeri dei tasti, ma il risultato ne vale la pena!
Per definire la musica è ancora necessario avere i numeri dei tasti. Questi sono riportati nella tavola qui sotto. Quando eseguite il programma, scrivete il contenuto di questa tabella, tre numeri alla volta.

R	Tasti	R	Tasti
1	12,12,19	33	19,16,12
2	12,12,21	34	21,16,12
3	14, 7,19	35	19,17,14
4	14, 7,17	36	17,17,14
5	16,12,16	37	16,19,16
6	16,12,16	38	16,19,16
7	12,12,12	39	12,19,12
8	12,12,12	40	12,19,12
9	12,12,19	41	19,16,12
10	12,12,21	42	21,16,12
11	14, 7,19	43	19,17,14
12	14, 7,17	44	17,17,14
13	16,12,16	45	16,19,16
14	16,12,16	46	16,19,16
15	12,12,12	47	12,19,12
16	12,12,12	48	12,19,12
17	16,12,12	49	12,19,16
18	16,12,12	50	12,21,16
19	17,14, 7	51	7,19,17
20	17,14, 7	52	7,17,17
21	19,16,12	53	12,16,19
22	19,16,12	54	12,16,19
23	19,12,12	55	12,12,19
24	19,12,12	56	12,12,19
25	16,12,12	57	12,19,16

26	16,12,12	58	12,21,16
27	17,14, 7	59	7,19,17
28	17,14, 7	60	7,17,17
29	19,16,12	61	12,16,19
30	19,16,12	62	12,16,19
31	19,12,12	63	12,12,19
32	19,12,12	64	12,12,19

Esempio 2 - Set di caratteri colorati

In questo esempio vogliamo semplicemente presentare un programma che mostra una parte dei caratteri ASCII in vari colori. Per terminare il programma premete un tasto qualsiasi e poi ENTER. Il programma è

```
100 CALL CLEAR
110 FOR C=1 TO 24
120 CALL HCHAR(C,3,64+C,28)
130 NEXT C
140 LET COLORE=10
150 FOR GRUPPO=5 TO 8
160 CALL COLOR(GRUPPO,COLORE
,16)
170 LET COLORE=COLORE+1
180 NEXT GRUPPO
190 INPUT A$
200 END
```

Esempio 3 - Caratteri grafici

Come ultimo esempio presenteremo un programma per disegnare sullo schermo una griglia formata da linee colorate. Eseguite il programma e state a guardare cosa succede. In seguito analizzate dettagliatamente il listato per capire il significato di ogni istruzione.

```
100 CALL CLEAR
110 LET A$="FFFFFFFFFFFFFFFF
"
120 CALL CHAR(96,A$)
130 LET Y=6
140 FOR X=6 TO 22 STEP 4
150 CALL VCHAR(Y,X,96,16)
160 NEXT X
170 LET X=6
180 FOR Y=6 TO 22 STEP 4
190 CALL HCHAR(Y,X,96,17)
200 NEXT Y
210 FOR COLORE=1 TO 16
220 CALL COLOR(9,COLORE,COLO
RE)
230 FOR T=1 TO 100
240 REM PAUSA
250 NEXT T
260 NEXT COLORE
270 END
```

PROBLEMI

1. Scrivete un programma per suonare la scala principale iniziando dal do centrale.

2. Scrivete un programma per suonare una canzone a vostra scelta.

3. Cosa accadrà se verrà eseguito il seguente programma?

```
100 LET A$="8080B0C88484C880
"
110 CALL CHAR(128,A$)
120 CALL CLEAR
130 CALL HCHAR(5,12,128,10)
140 END
```

4. Scrivete un programma per stampare LETTERE ROSSE sullo schermo usando caratteri rossi su uno sfondo bianco.

5. Disegnate le lettere minuscole a, b, c, d, e ed f. Usate il CALL CHAR per definire i caratteri ed inserirli nel computer. Scrivete un programma per visualizzare questi caratteri sullo schermo.

6. Disegnate un carattere con punti di color blu e sfondo trasparente su uno schermo bianco. Usate questo carattere per riempire ogni riga dello schermo.

7. Scrivete un programma per riempire lo schermo con il carattere H verde.

ESERCIZI

Affrontando questi esercizi potrete verificare la vostra preparazione riguardo gli argomenti di questo capitolo. Troverete le risposte alla fine del libro.

1. Cosa si ottiene con l'istruzione CALL SCREEN?

____

2. Nell'istruzione CALL HCHAR(Y,X,N,R), spiegate il significato di ogni variabile.

____

3. Qual è lo scopo del sottoprogramma CALL GCHAR?

____

4. Spiegate cosa si otterrà con CALL COLOR(6,11,16).

____

5. Qual è lo scopo del sottoprogramma CALL KEY?

____

6. Spiegate con precisione cosa si verifica con il comando CALL CLEAR.

____

7. Spiegate la funzione di ogni variabile nell'istruzione CALL SOUND(L,F,X).

____

Soluzioni degli esercizi

CAPITOLO 2

1. Premendo il tasto ENTER.

2. Premendo FNCT-Q (*Quit*). Oppure potete spegnere il computer e quindi riaccenderlo.

3. La moltiplicazione è contrassegnata dal simbolo *.

4. Scrivete CALL CLEAR e premete il tasto ENTER.

5. Il simbolo / indica la divisione.

6. Il computer mostrerà il numero 2 sullo schermo.

7. Sullo schermo si vedranno i caratteri 25/5+2.

8. Premere FNCT-S sette volte per spostare il cursore all'indietro sopra la G. Quindi scrivere la T e premere il tasto ENTER.

CAPITOLO 3

1. Premendo il tasto ENTER.

2. Premendo FNCT-4.

3. Premendo FNCT-4.

4. Verrà stampato il numero 1.

5. Per le variabili numeriche si possono usare fino a 15 caratteri e fino a 14 per le variabili di stringa (deve seguire il simbolo $).

6. Riscrivendo il numero di linea e premendo ENTER.

7. Semplicemente scrivendola usando un numero di linea non ancora presente nel programma.

8. Riscrivendola nella forma desiderata.

9. Scrivendo LIST e premendo il tasto ENTER.

10. Scrivendo CALL CLEAR e premendo ENTER.

11. Scrivendo NEW seguito da ENTER.

12. Scrivendo RUN e premendo il tasto ENTER.

13. Una variabile numerica si riferisce ad un numero. Una variabile di stringa denomina una serie di caratteri.

CAPITOLO 4

1. Gli operatori sono –, *, +, , e /.

2. Prima le potenze; poi le moltiplicazioni e le divisioni. Infine le addizioni e le sottrazioni.

3. Da sinistra a destra.

4. 100 LET A = (4+3*B/D)^2

5. Il numero 4.

6. a. 5.673E+14 b. 3.814275168E-06

7. a. 7258000 b. 0.001437

8. /, +, ^.

9. Scrivendo SAVE CS1 e seguendo le istruzioni.

10. Scrivendo OLD CS1 e seguendo le istruzioni.

CAPITOLO 5

---

1. Sullo schermo apparirà la sequenza di numeri riportata qui sotto.

```
1            2
3            4
5            6
7            8
     ecc.
```

2. a. con l'istruzione di assegnazione LET. b. INPUT. c. READ-DATA.

3. Una stringa.

4. Inserire note di spiegazione in un programma.

5. Una istruzione DATA.

6. Sullo schermo si vedrà Y = 3.

7. Due colonne per linea.

8. Quante si vuole.

9. Per organizzare la riga secondo la spaziatura voluta.

10. Si vedranno i numeri disposti come segue:

```
1            3
1  3
```

11. Il computer rileverà un errore nell'input poiché aspettava due nume-

ri mentre ne sono stati scritti tre. Il computer vi segnalerà di introdurre nuovamente i dati.

12.

```
100 PRINT "QUANTE MIGLIA";
110 INPUT M
120 LET K=1.609*M
130 PRINT M;"MIGLIA EQUIVALG
ONO A"
140 PRINT K;"KM"
150 END
```

CAPITOLO 6

1. Sullo schermo si vedrà la sequenza dei numeri 6, 10, 14 e 18.

2. Sullo schermo apparirà il messaggio

```
OTTIMO

DISTINTO
OTTIMO
BUONO
DISTINTO
OTTIMO

DATA ERROR IN 100
```

3.

```
100 PRINT "QUANTI ARTICOLI";
110 INPUT N
120 IF N<=20 THEN 160
130 IF N<=50 THEN 180
140 LET P=1.50
150 GOTO 190
160 LET P=2.00
170 GOTO 190
180 LET P=1.80
190 PRINT "IL PREZZO PER ART
ICOLO E'";P
200 LET C=N*P
210 PRINT "LA SPESA TOTALE E
'";C
220 GOTO 100
230 END
```

4.

```
100 LET NUMERO=0
110 PRINT NUMERO,
120 LET NUMERO=NUMERO+5
130 IF NUMERO<=115 THEN 110
140 END
```

5.

```
100 PRINT "LIMITE DI VELOCIT
A'";
110 INPUT L
```

```
120 PRINT "VELOCITA' EFFETTI
VA";
130 INPUT V
140 LET X=V-L
150 IF X<=10 THEN 210
160 IF X<=20 THEN 230
170 IF X<=30 THEN 250
180 IF X<=40 THEN 270
190 LET F=80000
200 GOTO 280
210 LET F=5000
220 GOTO 280
230 LET F=10000
240 GOTO 280
250 LET F=20000
260 GOTO 280
270 LET F=40000
280 PRINT "LA MULTA E' DI";F
;"LIRE"
290 END
```

CAPITOLO 7

1.

```
20            18
16            14
12            10
8             6
4             2
```

2. I numeri 1, 2, 3, 2, 4, 6, 3, 6, 9, 4, 9, 12 elencati in verticale.

3. a. 6, b. 7, c. 22.8, d. -1.

4. I cicli di I e J si intersecano.

5.

```
100 PRINT "MIGLIA","CHILOMET
RI"
110 PRINT
120 FOR M=10 TO 100 STEP 5
130 LET KM=1.609*M
140 PRINT M,KM
150 NEXT M
160 END
```

6.

```
100 READ N
110 LET SOMMA=0
120 FOR CONT=1 TO N
130 READ X
140 LET SOMMA=SOMMA+X
150 NEXT CONT
160 PRINT SOMMA/N
170 DATA 10
180 DATA 25,21,24,21,26,27,2
5,24,23,24
190 END
```

7. a. ABS(X) calcola il valore assoluto di X.

   b. SGN(X) calcola il segno algebrico di X. Se X è positivo, il valore della funzione è +1. Se X è negativo il valore della funzione è -1 e se X è 0 SGN(X) è 0.

   c. INT(X) è il primo intero minore di X.

   d. SQR(X) calcola la radice quadrata di X. X non può avere un valore negativo.

   e. SEG$ è usato per avere la parte di una stringa.

   f. VAL è usato per convertire un numero dalla rappresentazione in forma di stringa a quella numerica.

CAPITOLO 8

1. L'istruzione DIM si usa per riservare lo spazio in memoria per vettori e matrici sia numeriche che di stringa. L'istruzione OPTION stabilisce se il primo indice del vettore è 0 oppure 1.

2. X(3,4)

3. Il calcolatore stamperà la parola CARLO e il numero 183.

4.

```
100 OPTION BASE 0
110 DIM X(100)
120 PRINT "QUANTI NUMERI";
130 INPUT N
140 PRINT
150 PRINT " ","NUMERO"
160 PRINT
170 FOR I=1 TO N
190 PRINT I,
200 INPUT X(I)
210 NEXT I
220 LET S=0
230 FOR I=1 TO N
240 IF X(I)<0 THEN 260
250 LET S=S+X(I)
260 NEXT I
270 PRINT "LA SOMMA DEI NUME
RI"
280 PRINT "POSITIVI E'";S
290 END
```

5. X$(2,4)

6.

```
90 OPTION BASE 1
100 FOR RIGA=1 TO 4
110 FOR COL=1 TO 6
120 LET X(RIGA,COL)=4
130 NEXT COL
140 NEXT RIGA
150 FOR RIGA=1 TO 4
160 FOR COL=1 TO 6
170 PRINT X(RIGA,COL);
180 NEXT COL
190 PRINT
200 NEXT RIGA
210 END
```

7.

```
2   0   0   0   0
0   2   0   0   0
0   0   2   0   0
0   0   0   2   0
0   0   0   0   2
```

8. a. DIM A(2,3), b. 4, c. 3, e d. 3.

9. OPEN stabilisce una comunicazione tra il computer e un dispositivo esterno come ad esempio un registratore.

10. CLOSE interrompe la comunicazione stabilita dal comando OPEN.

CAPITOLO 9

1. a. 4, b. 14, c. 30, d. 80.

2.

```
2            1
3
4
```

3. a. GOSUB

   b. RETURN

   c. L'istruzione STOP equivale a GOTO a END.

4.

```
BIANCO
ROSSO
BLU
```

CAPITOLO 10

1.

```
100 RANDOMIZE
110 FOR CONT=1 TO 100
120 PRINT INT(4*RND+1),
130 NEXT CONT
140 END
```

2.

```
100 RANDOMIZE
110 FOR CONT=1 TO 100
120 PRINT 25*RND+25
130 NEXT CONT
140 END
```

3. Le parole BIANCO e ROSSO verranno scelte a caso e stampate dieci volte.

4. Cinque numeri compresi nell'intervallo da 0.00 a 9.99.

CAPITOLO 11

1. Questo comando renderà lo schermo giallo scuro.

2. Y definisce il numero di riga a partire dalla sommità dello schermo. X definisce il numero di colonna a partire da sinistra. N definisce il numero del carattere (codice ASCII) che viene stampato. R è un fattore di ripetizione.

3. Lo scopo del sottoprogramma GCHAR è quello di leggere il numero di codice ASCII del carattere che si trova sullo schermo alla riga e alla colonna specificate.

4. CALL COLOR (6,11,16) stabilisce il colore dei caratteri appartenenti al sottogruppo numero 6. I punti che formano i caratteri saranno giallo scuro e lo sfondo sarà bianco.

5. CALL KEY permette di interrogare la tastiera per sapere se viene premuto un tasto o per sapere che tasto è stato premuto dall'ultima esecuzione di una CALL KEY.

6. CALL CLEAR riempie lo schermo con il carattere di numero 32 di codice ASCII (*space*). Ciò pulisce lo schermo.

7. In CALL SOUND(L,F,X), L è la durata della nota in millisecondi, F è la frequenza in cicli al secondo, e X è l'attenuazione del volume in decibel.

Soluzioni dei problemi con numero dispari

CAPITOLO 5

1.

```
100 READ A,B,C,D
110 DATA 10,9,1,2
120 LET S=A+B
130 LET P=C*D
140 PRINT S,P
150 END
```

3. Il programma mostrerà 17 e 25 sulla stessa linea

5.

```
100 PRINT "TEMPO DI CADUTA (
SEC)";
110 INPUT T
120 LET DISTANZA=4.9*T^2
130 PRINT "L'OGGETTO PERCORR
E";DISTANZA;"METRI"
140 END
```

7.

```
100 INPUT A,B
110 LET T=B
120 LET B=A
130 LET A=T
140 PRINT A,B
150 END
```

9.

```
100 PRINT "CAPITALE";
110 INPUT P
120 PRINT "TASSO D'INT.(%)";
130 INPUT I
140 PRINT "PERIODO (ANNI)";
150 INPUT N
160 LET T=P*(1+I/100)^N
170 PRINT "LA SOMMA TOTALE E
'";T
180 END
```

CAPITOLO 6

1.

```
100 INPUT A,B
110 IF A>B THEN 140
120 PRINT B
130 GOTO 150
140 PRINT A
150 END
```

3.

```
100 LET SOMMA=0
110 LET NUMERO=0
120 LET SOMMA=SOMMA+NUMERO
130 LET NUMERO=NUMERO+1
140 IF NUMERO<=100 THEN 120
150 PRINT SOMMA
160 END
```

5. Il programma, una volta avviato, userà tutti i numeri (incluso 1111), e, dopo aver stampato DATA ERROR IN 120, si fermerà. La ragione è che il programma prevede una "variabile flag" 9999 che segni la fine dei dati, la quale non è presente. Così il programma legge più dati di quanti non siano presenti e si ferma.

7.

```
100 DATA 4,18,-3,-28,36,8
110 DATA 1,-6,12,9999
120 LET SOMMA=0
130 READ NUMERO
140 IF NUMERO=9999 THEN 190
150 IF NUMERO<-10 THEN 130
160 IF NUMERO>10 THEN 130
170 LET SOMMA=SOMMA+NUMERO
180 GOTO 130
190 PRINT SOMMA
200 END
```

9.

```
100 INPUT A,B
110 IF A<10 THEN 170
```

```
120 IF B<10 THEN 150
130 PRINT A+B
140 GOTO 210
150 PRINT A-B
160 GOTO 210
170 IF B>=10 THEN 200
180 PRINT A*B
190 GOTO 210
200 PRINT B-A
210 END
```

11.

```
100 PRINT "TASSO DI CRESCITA
 (%)";
110 INPUT R
120 LET N=0
130 LET Q=1
140 LET Q=Q*(1+R/100)
150 LET N=N+1
160 IF Q<2 THEN 140
170 PRINT "IL NUMERO DI PERI
ODI PER"
180 PRINT "RADDOPPIARE IL VA
LORE E'";N
190 END
```

CAPITOLO 7

1.

```
100 PRINT "N","SQR(N)"
110 PRINT
120 FOR N=2 TO 4 STEP .1
130 PRINT N,SQR(N)
140 NEXT N
150 END
```

3.

```
100 INPUT N
110 FOR X=2 TO N STEP 2
120 PRINT X
130 NEXT X
140 END
```

5. Sullo schermo appariranno i numeri 0, -1, 8, 0 e 0 disposti verticalmente in colonna.

7.

```
100 LET P=1
110 INPUT "FATTORIALE DI ":F
120 FOR LOOP=1 TO F
130 LET P=P*LOOP
140 NEXT LOOP
150 PRINT "IL FATTORIALE DI"
;F
160 PRINT "E' ";P
170 END
```

9. I cicli X e Z si incrociano.

11.

```
100 PRINT "INVESTIMENTO ANNU
ALE";
110 INPUT I
120 PRINT "TASSO DI INTERESS
E (%)";
130 INPUT R
140 PRINT "QUANTI ANNI ";
150 INPUT N
160 LET P1=0
170 FOR CTR=1 TO N
180 LET P2=(P1+I)*(1+R/100)
190 LET P1=P2
200 NEXT CTR
210 PRINT "ALLA FINE DELL'UL
TIMO ANNO"
220 PRINT "IL VALORE DELL'IN
VESTIMENTO"
230 PRINT "SARA' ";P1
240 END
```

13.

```
100 PRINT "STUDENTE","MEDIA
DEI VOTI"
110 PRINT
120 READ N
130 FOR CTR=1 TO N
140 READ S,V1,V2,V3
150 LET MEDIA=.25*V1+.25*V2+
.50*V3
160 PRINT S,MEDIA
170 NEXT CTR
180 DATA 6
190 DATA 3,90,85,92
200 DATA 1,75,80,71
210 DATA 6,100,82,81
220 DATA 5,40,55,43
230 DATA 2,60,71,68
240 DATA 4,38,47,42
250 END
```

15.

```
100 FOR X=1 TO 127
110 PRINT CHR$(X)
120 NEXT X
130 END
```

CAPITOLO 8

1.

```
100 OPTION BASE 1
110 DIM X(25)
120 INPUT N
130 FOR I=1 TO N
140 READ X(I)
150 NEXT I
160 FOR I=1 TO N
```

```
170 PRINT X(I);
180 NEXT I
190 DATA 12
200 DATA 2,1,4,3,2,4,5,6,3,5
,4,1
210 END
```

3. Sullo schermo apparirà il numero 10.

5.

```
100 OPTION BASE 1
110 DIM X(100)
120 INPUT N
130 FOR I=1 TO N
140 PRINT I;
150 INPUT X(I)
160 NEXT I
170 FOR I=1 TO N-1
180 IF X(I+1)<=X(I)THEN 230
190 LET TEMP=X(I)
200 LET X(I)=X(I+1)
210 LET X(I+1)=TEMP
220 GOTO 170
230 NEXT I
240 FOR I=1 TO N
250 PRINT X(I)
260 NEXT I
270 END
```

7.

```
1  1  1  1  1  1

0  0  0  0  0  0

0  0  1  1  1  1

0  0  0  0  0  0

0  0  0  0  1  1

0  0  0  0  0  0
```

9.

```
100 OPTION BASE 1
110 DIM X(2,5)
120 FOR RIGA=1 TO 2
130 FOR COL=1 TO 5
140 READ X(RIGA,COL)
150 NEXT COL
160 NEXT RIGA
170 DATA 2,1,0,5,1
180 DATA 3,2,1,3,1
190 FOR RIGA=1 TO 2
200 FOR COL=1 TO 5
210 PRINT X(RIGA,COL);
220 NEXT COL
230 PRINT
240 PRINT
250 NEXT RIGA
260 END
```

11.

```
100 OPTION BASE 1
110 DIM A(30,30)
120 PRINT "QUANTE RIGHE";
130 INPUT R
140 PRINT "QUANTE COLONNE";
150 INPUT C
160 FOR R1=1 TO R
170 FOR C1=1 TO C
180 PRINT "RIGA";R1;"COL";C1
;
190 INPUT A(R1,C1)
200 NEXT C1
210 NEXT R1
220 FOR R1=1 TO R
230 LET S=0
240 FOR C1=1 TO C
250 LET S=S+A(R1,C1)
260 NEXT C1
270 PRINT "LA SOMMA DI RIGA"
;R1;"E'";S
280 NEXT R1
290 FOR C1=1 TO C
300 LET P=1
310 FOR R1=1 TO R
320 LET P=P*A(R1,C1)
330 NEXT R1
340 PRINT "IL PRODOTTO DELLA
 COLONNA";C1;"E'";P
350 NEXT C1
360 END
```

13.

```
100 OPTION BASE 1
110 DIM VENDITE(4,6),TOTGIOR
(6),TOTSETT(4)
120 FOR COMM=1 TO 4
130 PRINT "VENDITE GIORNALIE
RE TOTALI"
140 PRINT "DEL COMMESSO";COM
M
150 FOR GIORNO=1 TO 6
160 PRINT "GIORNO";GIORNO
170 INPUT VENDITE(COMM,GIORN
O)
180 NEXT GIORNO
190 NEXT COMM
200 LET TOT=0
210 FOR COMM=1 TO 4
220 LET TOTGIOR(COMM)=0
230 FOR GIORNO=1 TO 6
240 LET TOTGIOR(GIORNO)=TOTG
IOR(GIORNO)+VENDITE(COMM,GIO
RNO)
250 LET TOTSETT(COMM)=TOTSET
T(COMM)+VENDITE(COMM,GIORNO)
260 LET TOT=TOT+VENDITE(COMM
,GIORNO)
270 NEXT GIORNO
280 NEXT COMM
290 PRINT
300 PRINT "COMMESSO","TOTALE
 SETT."
310 FOR COMM=1 TO 4
320 PRINT COMM,TOTSETT(COMM)
330 NEXT COMM
340 PRINT
350 PRINT "GIORNO","TOTALE G
IOR."
360 FOR GIORNO=1 TO 6
370 PRINT GIORNO,TOTGIOR(GIO
RNO)
380 NEXT GIORNO
```

```
390 PRINT
400 PRINT "IL TOTALE DELLE V
ENDITE"
410 PRINT "SETTIMANALI E' ";
TOT
420 END
```

15.

```
100 OPTION BASE 1
110 DIM NOME$(20),VOTO(20)
120 PRINT "QUANTI SONO I NOM
I";
130 INPUT N
140 PRINT
150 PRINT "SCRIVI I NOMI E I
 VOTI      SEPARATI DA UNA V
IRGOLA"
160 FOR I=1 TO N
170 INPUT NOME$(I),VOTO(I)
180 NEXT I
190 PRINT
200 FOR I=1 TO N-1
210 IF VOTO(I+1)<VOTO(I)THEN
 290
220 LET TEMP=VOTO(I)
230 LET TEMP$=NOME$(I)
240 LET VOTO(I)=VOTO(I+1)
250 LET VOTO(I+1)=TEMP
260 LET NOME$(I)=NOME$(I+1)
270 LET NOME$(I+1)=TEMP$
280 GOTO 200
290 NEXT I
300 PRINT "VOTO","NOME"
310 PRINT
320 FOR I=1 TO N
330 PRINT VOTO(I),NOME$(I)
340 NEXT I
350 END
```

17.

```
100 OPTION BASE 1
110 DIM NOME$(10)
120 OPEN #1:"CS1",INPUT ,FIX
ED 64
130 FOR I=1 TO 10
140 INPUT #1:NOME$(I)
150 NEXT I
160 FOR I=1 TO 9
170 IF NOME$(I)<NOME$(I+1)TH
EN 220
180 LET TEMP$=NOME$(I)
190 LET NOME$(I)=NOME$(I+1)
200 LET NOME$(I+1)=TEMP$
210 GOTO 160
220 NEXT I
230 FOR I=1 TO 10
240 PRINT NOME$(I)
250 NEXT I
260 CLOSE #1
270 END
```

CAPITOLO 9

1.

```
25                    5
20
65
```

3.

```
2    7    1    3    3
7    10   3    3    3
10   3    3    8    8
```

5.

```
100 REM  SUBROUTINE
110 LET T=0
120 FOR I=1 TO Z(0)
130 LET T=T+Z(I)
140 NEXT I
150 RETURN
```

7.

```
100 OPEN #1:"CS1",INPUT ,FIX
ED 51
110 INPUT #1:TOTALE
120 FOR CTR=1 TO TOTALE
130 INPUT #1:A$
140 PRINT "STANZA: ";SEG$(A$
,2,15)
150 PRINT "ARTICOLO: ";SEG$(
A$,17,15)
160 PRINT "ACQUISTATO NEL: 1
9";SEG$(A$,32,2)
170 PRINT "PREZZO INIZIALE:
L.";SEG$(A$,31,9)
180 PRINT "VALORE CORRENTE:
L.";SEG$(A$,43,9)
190 PRINT
200 NEXT CTR
210 END
```

CAPITOLO 10

1.

```
100 RANDOMIZE
110 FOR CTR=1 TO 25
120 PRINT INT(100*RND)/10
130 NEXT CTR
140 END
```

3. Sullo schermo si vedrà una serie di venti numeri casuali compresi tra 0.01 e 0.20.

5.

```
100 RANDOMIZE
110 FOR I=1 TO 5
120 READ N
130 LET TESTA=0
140 LET CROCE=0
150 FOR CTR=1 TO N
160 LET X=INT(2*RND+1)
170 IF X=1 THEN 200
180 LET CROCE=CROCE+1
190 GOTO 210
200 LET TESTA=TESTA+1
210 NEXT CTR
220 PRINT
230 PRINT "DOPO";N;"LANCI RI
SULTA"
240 PRINT TESTA;"VOLTE TESTA
"
250 PRINT CROCE;"VOLTE CROCE
"
260 NEXT I
270 DATA 10,50,100,500,1000
280 END
```

7.

```
100 RANDOMIZE
110 LET SOMMA=0
120 FOR CTR=1 TO 1000
130 LET SOMMA=SOMMA+RND
140 NEXT CTR
150 LET MEDIA=SOMMA/1000
160 PRINT MEDIA
170 END
```

9.

```
100 RANDOMIZE
110 LET INCONTRO=0
120 FOR CTR=1 TO 1000
130 LET GIOVANNI=60*RND
140 LET GIORGIO=60*RND
150 IF ABS(GIOVANNI-GIORGIO)
>10 THEN 170
160 LET INCONTRO=INCONTRO+1
170 NEXT CTR
180 PRINT "LA PROBABILITA' C
HE SI       INCONTRINO E'";IN
CONTRO/1000
190 END
```

11.

```
100 RANDOMIZE
110 FOR LOOP=1 TO 25
120 LET SOMMA=0
130 FOR CTR=1 TO 12
140 LET SOMMA=SOMMA+RND
150 NEXT CTR
160 LET R=10+2*(SOMMA-6)
170 PRINT INT(100*R+.5)/100
180 NEXT LOOP
190 END
```

CAPITOLO 11

1.

```
100 DATA 264,296,334,352
110 DATA 396,444,499,528
120 FOR I=1 TO 8
130 READ FREQ
140 CALL SOUND(1000,FREQ,0)
150 NEXT I
160 END
```

3. Verrà stampata orizzontalmente una stringa formata da 10 lettere b minuscole a partire dall'incrocio di riga 5 e colonna 12.

5.

```
100 DATA "00001A264242261A"
110 DATA "4040586442426458"
120 DATA "00001C224040221C"
130 DATA "02021A264242261A"
140 DATA "00001C227E40221C"
150 DATA "0008142470202020"
160 LET PAROLE$=" "
170 FOR CTR=128 TO 133
180 READ A$
190 CALL CHAR(CTR,A$)
200 LET PAROLA$=PAROLA$&CHR$
(CTR)
210 NEXT CTR
220 CALL CLEAR
230 PRINT PAROLA$
240 END
```

7.

```
100 CALL CLEAR
110 CALL SCREEN(16)
120 CALL COLOR(6,3,1)
130 CALL HCHAR(1,1,72,768)
140 FOR T=1 TO 5000
150 REM PAUSA
160 NEXT T
170 END
```

Finito di stampare il 10 novembre 1983 presso Lito Velox - Trento
Fotocomposto e impaginato da Lito Velox - Trento
Printed in Italy

Siete interessati ai personal computer?

Su questo argomento, nella collana "il piacere del computer" sono stati pubblicati tra gli altri i seguenti titoli.

32 programmi con il PET
Il volume contiene 32 programmi in Basic, completamente documentati con listati, esecuzioni di prova, istruzioni per far girare il programma, suggerimenti per variazioni, ecc. Tutti i programmi sono stati verificati e possono essere eseguiti su ogni tipo di PET.
240 pagine, 16.000 lire, sigla PDC 1

Intervista sul personal computer, hardware
Seicento domande e risposte sul mondo dei personal computer. Questo primo volume, dedicato all'hardware, contiene una introduzione, sotto forma di intervista, ai computer in generale e ai microprocessori in particolare.
240 pagine, 12.000 lire, sigla PDC 2

32 programmi con l'Apple
Il volume contiene 32 programmi in Basic, completamente documentati con listati, esecuzioni di prova, istruzioni per far girare il programma, suggerimenti per variazioni, ecc. Tutti i programmi sono stati verificati e possono essere eseguiti su ogni tipo di Apple II.
248 pagine, 12.000 lire, sigla PDC 3

Microsoft Basic
Un breve manuale di introduzione al Microsoft Basic, il Basic dei personal computer. Oggi il Microsoft Basic, una evoluzione e specializzazione del Basic originale, è di fatto lo standard del Basic per microcomputer.
150 pagine, 12.000 lire, sigla PDC 4

Pascal
Questo testo è scritto per coloro che non hanno esperienza di calcolatori o programmazione. Gli argomenti sono organizzati in modo che il lettore possa iniziare a programmare fin dall'inizio.
200 pagine, 12.000 lire, sigla PDC 5

32 programmi con il TRS-80
Trentadue programmi, completamente documentati, pronti per essere eseguiti su un TRS-80 modello I. Il volume comprende i listati, le spiegazioni e i consigli per ulteriori progetti.
240 pagine, 12.000 lire, sigla PDC 6

Intervista sul personal computer, software
In questo secondo volume, dedicato al software, altre centinaia di domande e risposte sul mondo dei personal computer. Contiene una introduzione alla programmazione, ai linguaggi assembler e a quelli evoluti.
200 pagine, 12.000 lire, sigla PDC 7

Imparate il Basic con il PET/CBM
Questo libro è stato progettato per essere utile a chiunque desideri imparare a programmare in Basic avendo a disposizione un PET.
250 pagine, 16.000 lire, sigla PDC 8

Il personal computer come professione
Il personal computer vi offre mille opportunità di lavoro: potete scrivere articoli o libri su computer; intraprendere un'attività di consulenza o organizzare un'esposizione locale. Per ognuna di queste attività e per molte altre il libro contiene numerosi consigli per la migliore riuscita.
112 pagine, 9.000 lire, sigla PDC 9

Te ne intendi di computer?
Lo scopo di questo libro è di aumentare il livello della vostra comprensione dei computer. Sapere cosa possono e cosa non possono fare. Sapere qual è il loro ruolo nella società, sapere quali problemi creano.
144 pagine, 12.000 lire, sigla PDC 10

Troverete questi libri nelle principali librerie, oppure potete ordinarli direttamente alla casa editrice compilando la cartolina qui allegata

SUL RETRO ALTRE INFORMAZIONI

Ho trovato questa cartolina nel libro ....

acquistato in ....

In questo spazio potete scrivere quello che pensate di questo libro.

☐ Desidero ricevere il vostro più recente catalogo

☐ Desidero ricevere i volumi qui indicati:

sigla	titolo	prezzo
	totale	

☐ Pagherò al postino l'importo totale indicato + L. 1.000 quale contributo alle spese di spedizione

☐ Allego assegno o vaglia n. .... per l'importo totale indicato

Il Basic e il personal computer, uno: introduzione
Il libro, scritto in tono amichevole e informale, non richiede precedenti esperienze con i computer. Vi è compresa una presentazione del Basic con decine di esempi dettagliati, che fanno diventare realtà le vostre idee.
190 pagine, 14.000 lire, sigla PDC 11

Imparate il linguaggio dell'Apple
Se conoscete già il Basic e possedete un computer Apple, siete sulla strada giusta per diventare un esperto in materia. Questo libro vi mette in grado di comprendere il linguaggio macchina dell'Apple partendo dagli esempi più semplici ed arrivando all'uso del miniassembler.
340 pagine, 15.000 lire, sigla PDC 12

Il Basic e il personal computer, due: applicazioni
Questo volume contiene numerosi programmi in Basic adatti ad ogni personal computer. L'uso delle variabili alfanumeriche, gli algoritmi di sort, i giochi, l'arte con il computer sono esempi di programmi illustrati nel volume.
200 pagine, 14.000 lire, sigla PDC 13

Il manuale del CP/M
Il CP/M è il sistema operativo maggiormente usato per i personal computer. Questo volume è un'introduzione breve ma efficace alla tecnica e alla filosofia del CP/M.
120 pagine, 9.500 lire, sigla PDC 14

A scuola con il PET/CBM
Trenta programmi didattici per ogni tipo di PET/CBM matematica, fisica, statistica, ecc.
160 pagine, 13.000 lire, sigla PDC 15

Il manuale dell'Atom
L'edizione italiana del manuale originale di questo personal computer, ora importato in Italia.
300 pagine, 18.500 lire, sigla PDC 16

Il libro del Commodore VIC 20
Questo libro è per i possessori, i futuri possessori e gli utenti di un personal computer VIC Commodore. È inteso come supplemento al manuale della macchina, e presenta numerose caratteristiche del VIC.
156 pagine, 12.000 lire, sigla PDC 17

Il debug nei personal computer
Il "debug", in informatica, è la messa a punto dei programmi, la ricerca e la correzione finale degli errori. Si tratta di una operazione di grande importanza, che si può effettuare in vari modi, tutti trattati in questo libro.
144 pagine, 15.000 lire, sigla PDC 18

Programmazione in Basic per l'uomo d'affari
Questo libro è per gli uomini d'affari che vogliono imparare ad usare il calcolatore. Il libro comincia con la convinzione che sia più facile per un uomo d'affari imparare a programmare, che per un programmatore imparare l'organizzazione e la gestione di una azienda.
256 pagine, 19.000 lire, sigla PDC 19

Imparate il Basic con lo ZX-81
È il microcomputer più venduto nel mondo. Ma come ogni computer, necessita di software e documentazione: questo libro contiene 37 programmi che illustrano tutte le caratteristiche e le capacità della macchina.
132 pagine, 13.000 lire, sigla PDC 20

Dal Basic al Pascal
Questa eccezionale guida rende facile per chiunque passare dalla propria conoscenza del Basic ad una perfetta padronanza del Pascal, un linguaggio a 2 livelli che consente di accelerare il tempo reale di operazione e di controllo della programmazione e che richiede uno spazio inferiore di memoria nel vostro computer.
320 pagine, 19.000 lire, sigla PDC 21

Imparate il Basic con il Texas TI 99/4A
Il Texas TI 99/4A vi può aiutare nell'apprendimento delle lingue, della matematica, giocare al videogame, tenere la contabilità della vostra famiglia. Avete bisogno solo di conoscere il linguaggio giusto: questo libro ve lo insegna.
264 pagine, 22.000 lire, sigla PDC 22

A scuola con il Texas TI 99/4A
Scritto da un isegnante, questo è un libro di software per la scuola. Gli argomenti sono presentati in ordine di difficoltà crescente e sono tutti generalmente trattati negli studi medi inferiori e superiori. Ma non mancano applicazioni varie e giochi.
212 pagine, 18.000 lire, sigla PDC 23

cognome e nome

indirizzo

cap. località

(partita iva o codice fiscale)

firma

il formato della cartolina è conforme alle vigenti norme postali

cedola di commissione libraria

franco muzzio editore

via bonporti, 36

35141 padova

Questa guida vi introduce nel mondo emozionante dei personal computer.

il Texas TI 99/4A vi può aiutare nell'apprendimento delle lingue, della matematica, vi potrà far giocare ai migliori videogame, potrà tenere la contabilità della vostra famiglia.
Avete bisogno solo di conoscere il linguaggio giusto: questo libro ve lo insegna.

franco muzzio & c. editore

L. 22.000 (21.568) ISBN 88-7021-237-8